Jean-Baptiste Civet
Boris Hanuš

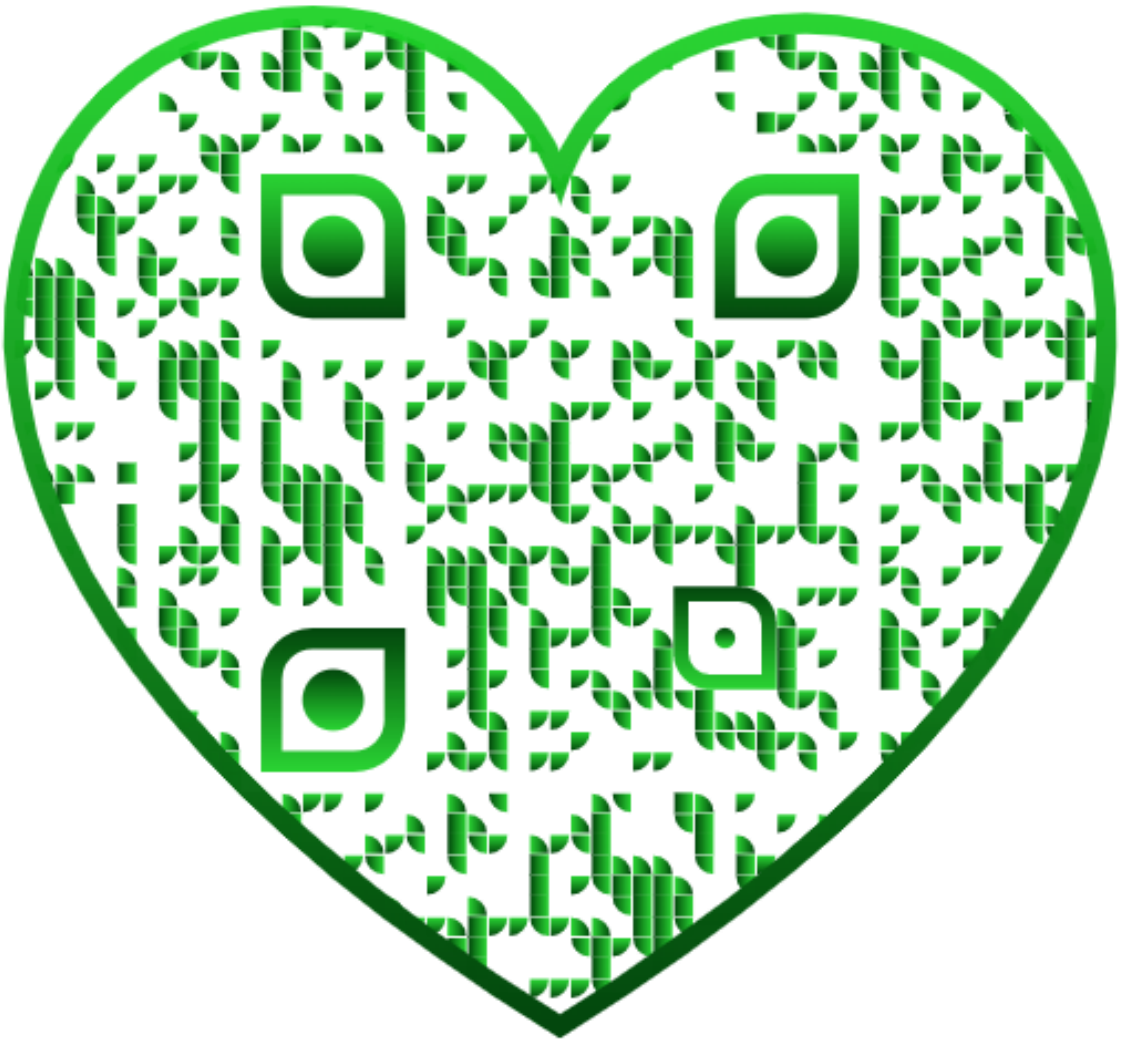
2nde ICN

Algorithmique et programmation avec la TI-83 Premium CE

Conception de jeux et projets avec le TI-Innovator™ Hub

EYROLLES

TEXAS INSTRUMENTS



Résumé

Un apprentissage ludique de la programmation

Grâce à différents projets de difficulté progressive, ce cahier d'activités vous initiera à la pratique de l'algorithmique avec la calculatrice TI-83 Premium CE, en explorant deux champs complémentaires de la programmation. Vous apprendrez d'une part à générer des images et des graphismes avec votre machine pour réaliser un labyrinthe et un jeu de basket. Vous utiliserez d'autre part le TI-Innovator™ Hub, un boîtier capable d'interagir avec son environnement au moyen de capteurs et d'actionneurs, afin de concevoir un piano téléguidé et un ascenseur.

Issus de projets réalisés avec des élèves de seconde, les chapitres de ce livre comportent un grand nombre d'exercices, de petits défis, ainsi que des programmes à exécuter ou à compléter. Toutes les solutions de ces exercices sont disponibles sur *http://go.eyrolles.com/ti* ou par les QR codes figurant dans l'ouvrage. Vous y trouverez également des vidéos qui viendront compléter certaines explications théoriques ou pratiques.

Au sommaire

Préambule : programmation avec la TI-83 Premium CE et le TI-Innovator™ Hub. Écriture d'un programme • Programmation graphique • Programmation avec le TI-Innovator™ Hub • **Le labyrinthe.** Les codes des touches • Affichage d'un point à l'écran • Affichage et déplacement du personage • Construction du labyrinthe et finalisation du jeu • **Le piano numérique.** Et si on parlait couleurs ? Transformer sa calculatrice en piano • Le capteur de lumière • Le capteur de distance • **Le jeu de basket.** Préparation des images d'arrière-plan • Description du jeu et tracé des dessins • Codage de la boucle principale du jeu • **L'ascenseur.** Matériel nécessaire au projet • Connexion des microrupteurs • Connexion du relais et du moteur • Construction de la cage d'ascenseur • Branchement du circuit et écriture du code • Gestion de la descente • **Annexe : liens utiles.**

À qui s'adresse ce livre ?

- Aux élèves de seconde ICN
- À tous ceux qui souhaitent progresser en programmation

Biographie auteur

Passionné par la robotique et les nouvelles technologies, **Jean-Baptiste Civet** est professeur de mathématiques et membre de l'équipe T3 *(Teachers Teaching with Technology)*, un réseau international d'enseignants fédéré par Texas Instruments. Il participe à la formation continue (mathématiques, usage du numérique) et anime un atelier de robotique dans son établissement depuis sept ans.

Professeur de mathématiques en lycée et dans le supérieur, également membre de l'équipe T3, **Boris Hanuš** enseigne les spécialités ISN en terminale S et ICN en seconde depuis leur création. Très investi dans l'utilisation de l'informatique, des calculatrices et du calcul formel avec ses élèves, il contribue au développement de l'usage des nouvelles technologies dans son établissement.

www.editions-eyrolles.com

Jean-Baptiste Civet
Boris Hanuš

Algorithmique et **programmation** avec la **TI-83** Premium CE

EYROLLES

TEXAS INSTRUMENTS

Éditions Eyrolles
61, bd Saint-Germain
75005 Paris
www.editions-eyrolles.com

Merci à Carlos, Johanna et Antoine pour leur soutien dans ce projet.

Attention : la version originale de cet ebook est en couleur, lire ce livre numérique sur un support de lecture noir et blanc peut en réduire la pertinence et la compréhension.

Attention : pour lire les exemples de lignes de code, réduisez la police de votre support au maximum.

Conception graphique et mise en pages : Nord Compo

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre français d'exploitation du droit de copie, 20, rue des Grands-Augustins, 75006 Paris.

© A4 Technologie pour les photos des pages 60 et 61
© Jean-Baptiste Civet pour les photos des pages 55 et 62
© Texas Instruments pour toutes les autres photos de l'ouvrage

© Groupe Eyrolles, 2017
ISBN : 978-2-212-67400-2

Avant-propos

Cet ouvrage à vocation pratique va vous faire découvrir deux champs complémentaires de l'algorithmique et de la programmation avec la TI-83 Premium CE. D'une part, vous apprendrez à créer des images et des graphismes sur votre calculatrice dans le but de concevoir un labyrinthe et un jeu de basket. D'autre part, vous utiliserez le TI-Innovator™ Hub, un boîtier relié à la calculatrice qui est capable de communiquer et d'interagir avec son environnement à l'aide de capteurs et d'actionneurs. Vous serez ainsi amené à construire un piano téléguidé et un ascenseur, ni plus ni moins !

Chaque chapitre est ponctué par de nombreux exercices et défis à réaliser. À côté de chaque énoncé d'exercice, vous trouverez un QR code et son lien associé pointant tous deux sur la solution. Ce lien est présenté de manière réduite (`go.eyrolles.com/...`) mais doit en réalité s'écrire sous la forme `http://go.eyrolles.com/...` La solution pourra être un programme à télécharger, ou/et un fichier PDF à lire, ou/et une vidéo à visionner. Dans ce dernier cas, le QR code sera accompagné du symbole .

Tous ces fichiers et vidéos sont par ailleurs disponibles sur l'extension web de l'ouvrage `http://go.eyrolles.com/ti`, sur laquelle figurent également les solutions de chaque défi.

À vous de jouer maintenant !

Sommaire

Préambule

Programmation avec la TI-83 Premium CE et le TI-Innovator™ Hub

Écriture d’un programme

En utilisant l’unité nomade (la calculatrice)
En utilisant le logiciel TI Connect™ CE

Programmation graphique

Fenêtrage
Primitives de dessin

Programmation avec le TI-Innovator™ Hub

Installons l’application TI-Innovator™ Hub
Allumons une LED
Ouvrons le TI-Innovator™ Hub à son environnement

Chapitre 1

Le labyrinthe

ÉTAPE 1 **Les codes des touches**

ÉTAPE 2 **Affichage d’un point à l’écran**

En utilisant les coordonnées du repère
En utilisant les coordonnées des pixels

ÉTAPE 3 **Affichage et déplacement du personnage**

ÉTAPE 4 **Construction du labyrinthe et finalisation du jeu**

Chapitre 2

Le piano numérique

ÉTAPE 1 **Et si on parlait couleurs ?**

Sur la TI-83 Premium CE

Sur le TI-Innovator™ Hub

ÉTAPE 2 **Transformer sa calculatrice en piano**

L'instruction SOUND

La leçon de piano

ÉTAPE 3 **Le capteur de lumière**

ÉTAPE 4 **Le capteur de distance**

Chapitre 3

Le jeu de basket

ÉTAPE 1 **Préparation des images d'arrière-plan**

ÉTAPE 2 **Description du jeu et tracé des dessins**

Position de la main

Le panier de basket et les jauges

Test intermédiaire

ÉTAPE 3 **Codage de la boucle principale du jeu**

Chapitre 4

L’ascenseur

Matériel nécessaire au projet

ÉTAPE 1 Connexion des microrupteurs

ÉTAPE 2 Connexion du relais et du moteur

ÉTAPE 3 Construction de la cage d’ascenseur

ÉTAPE 4 Branchement du circuit et écriture du code

ÉTAPE 5 Gestion de la descente

Annexe

Liens utiles

Préambule

Programmation avec la TI-83 Premium CE et le TI-Innovator™ Hub

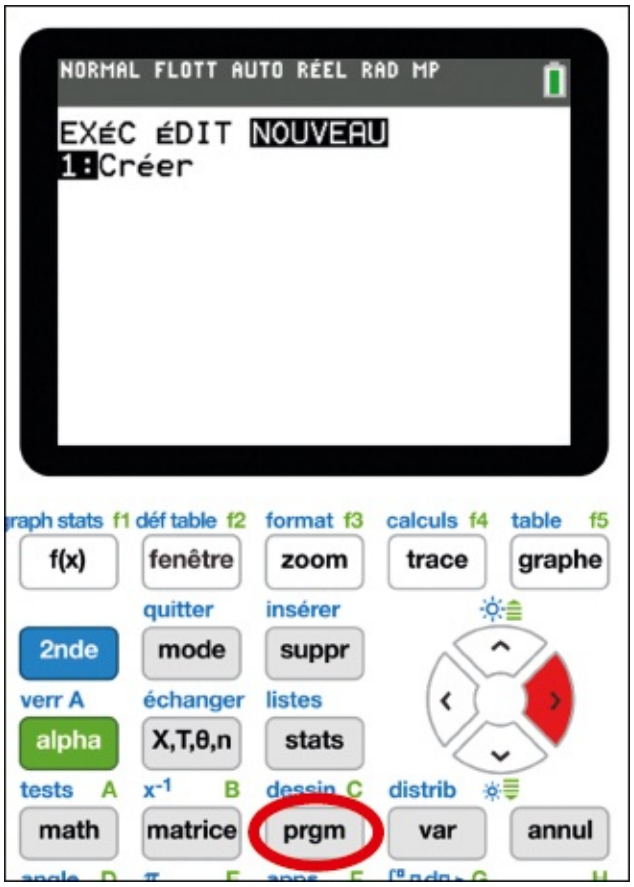
Avant de découvrir les différents projets pratiques de ce livre, étudions ensemble quelques manipulations de la TI-83 Premium CE qui vont être essentielles pour la suite. Nous en profiterons également pour faire connaissance avec le TI-Innovator™ Hub, un boîtier qui permet d'étendre les fonctionnalités de la calculatrice.

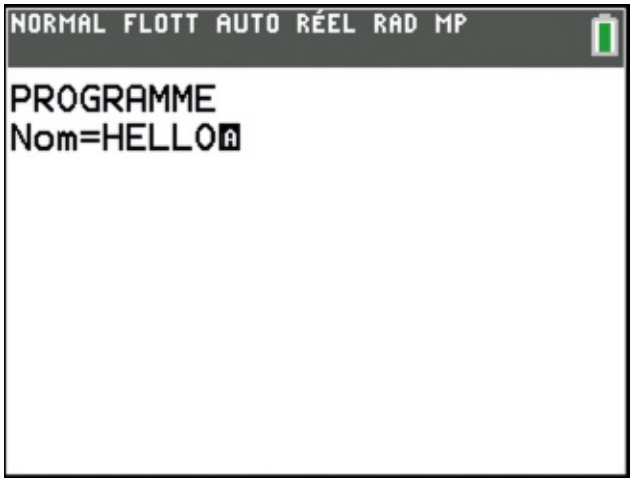
Écriture d’un programme

Pour écrire un programme avec la TI-83 Premium CE, vous avez deux possibilités : soit utiliser directement le clavier de la calculatrice, soit saisir ce programme sur ordinateur et le transférer sur la calculatrice grâce au logiciel TI Connect™ CE.

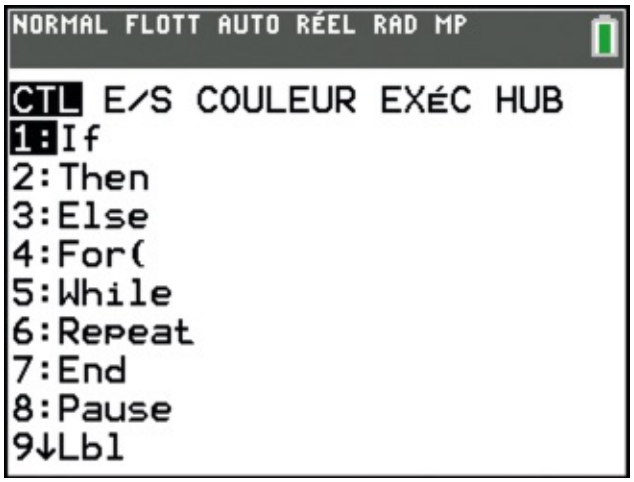
En utilisant l’unité nomade (la calculatrice)

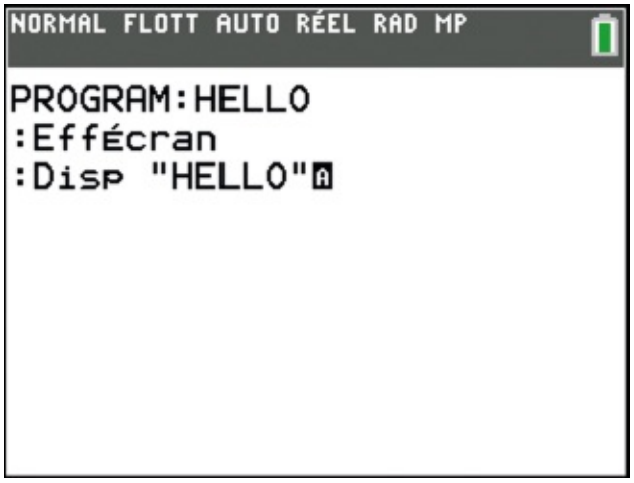
Appuyez d’abord sur la touche [prgm] de la TI-83 Premium CE. Puis, à l’aide des flèches, allez dans l’onglet `NOUVEAU`, choisissez la seule action possible `Créer` et validez votre choix via la touche [entrer]. Saisissez alors le nom de votre programme. Ce n’est qu’une fois cette opération de saisie validée par la touche [entrer] que vous pourrez commencer à écrire le code du programme.


NORMAL FLOTT AUTO RÉEL RAD MP
EXéC éDIT NOUVEAU
1:Créer
graph stats f1 déf table f2 format f3 calculs f4 table f5
f(x) fenêtre zoom trace graphe
quitter insérer
2nde mode suppr
verr A échanger listes
alpha X,T,θ,n stats
tests A x-1 B dessin C distrib
math matrice prgm var annul

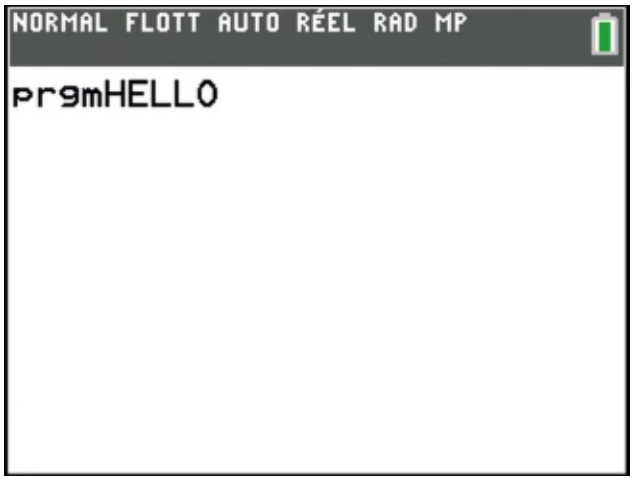


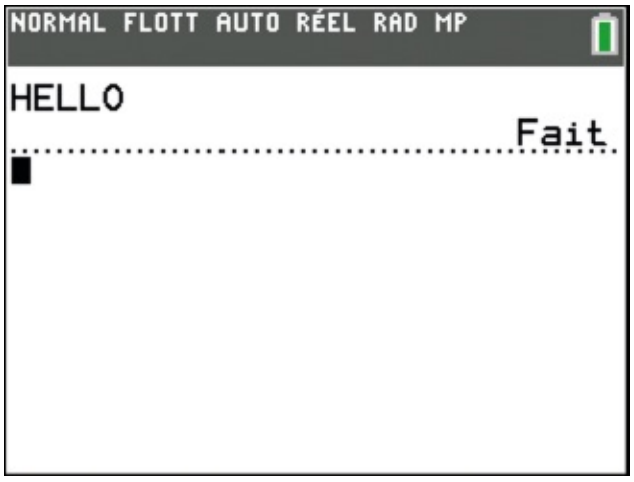
Vous remarquerez que si vous appuyez de nouveau sur la touche [prgm], le menu a changé et donne désormais accès aux fonctions de programmation classiques : boucles, conditions, affichages... Saisissez alors votre programme. Enfin, quittez l'éditeur en appuyant sur [2nde], [quitter] [mode].





Pour exécuter le programme, appuyez sur la touche [prgm], sélectionnez-le dans la liste des programmes proposés et validez votre choix via la touche [entrer]. Le programme efface l'écran, écrit « Hello », puis s'arrête.

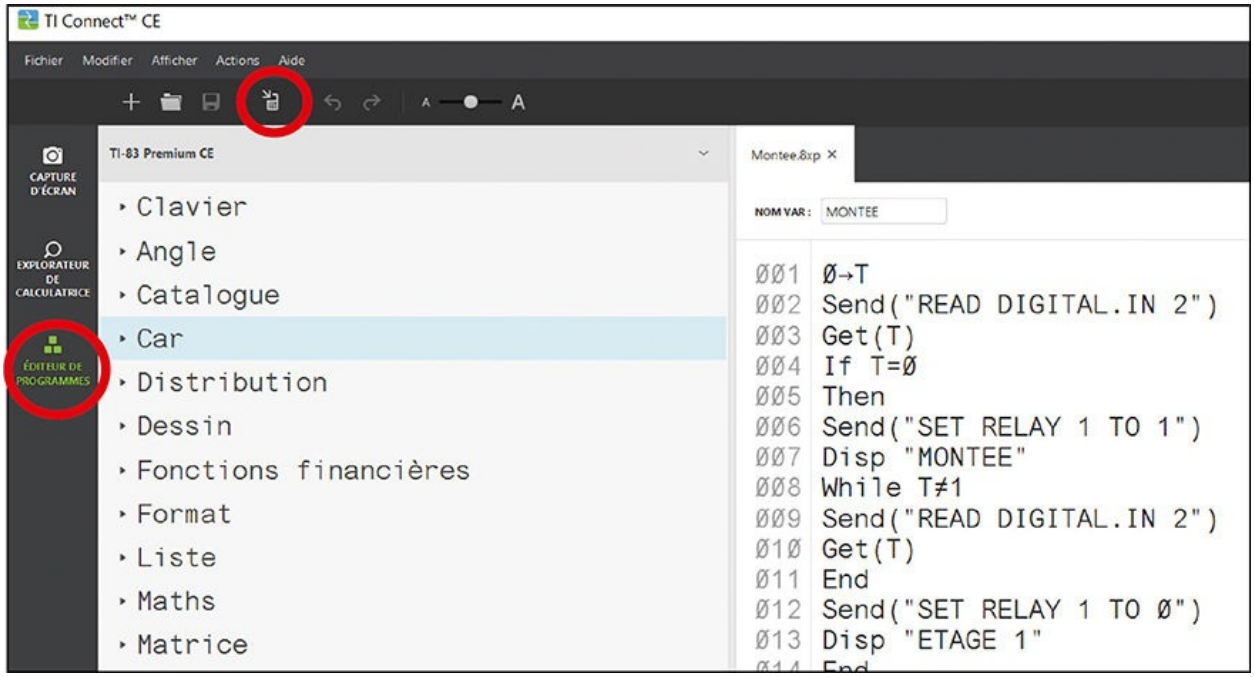




En utilisant le logiciel TI Connect™ CE

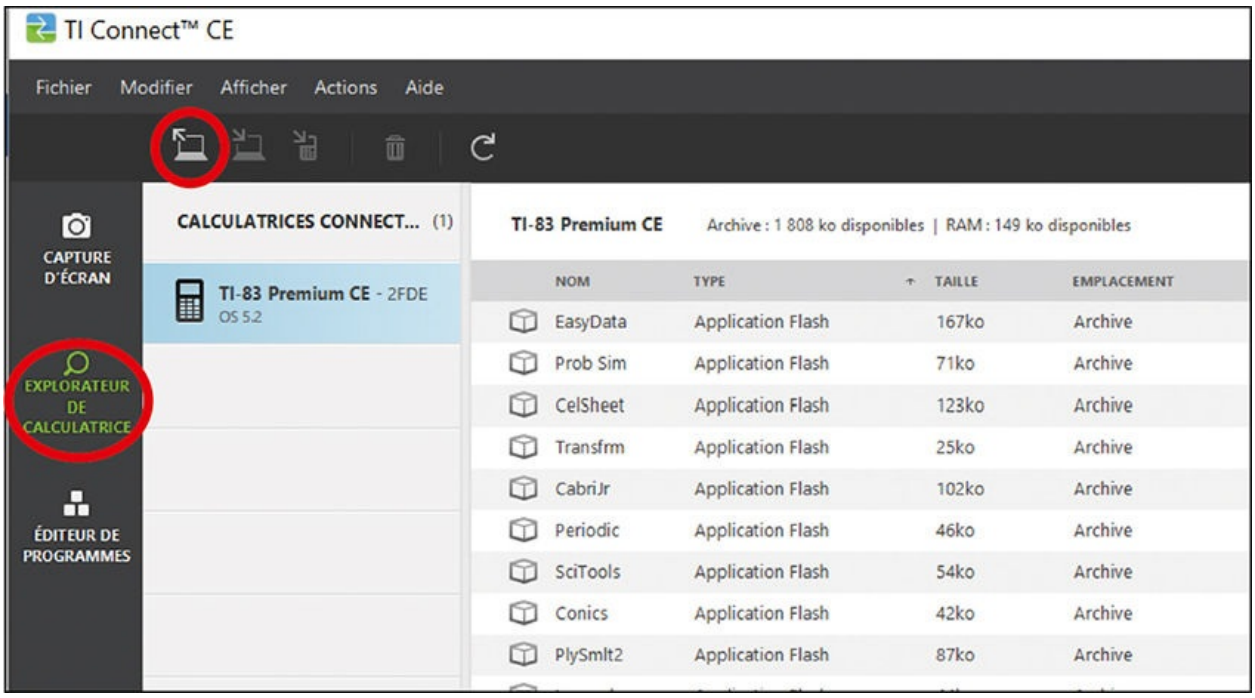
Ce logiciel pour PC et Mac peut être téléchargé gratuitement sur le site de Texas Instruments : **`https://education.ti.com/fr/products/computer-software/ti-connect-ce-sw`**. Il permet notamment de saisir des programmes sur votre ordinateur et de les transférer sur la TI-83 Premium CE.

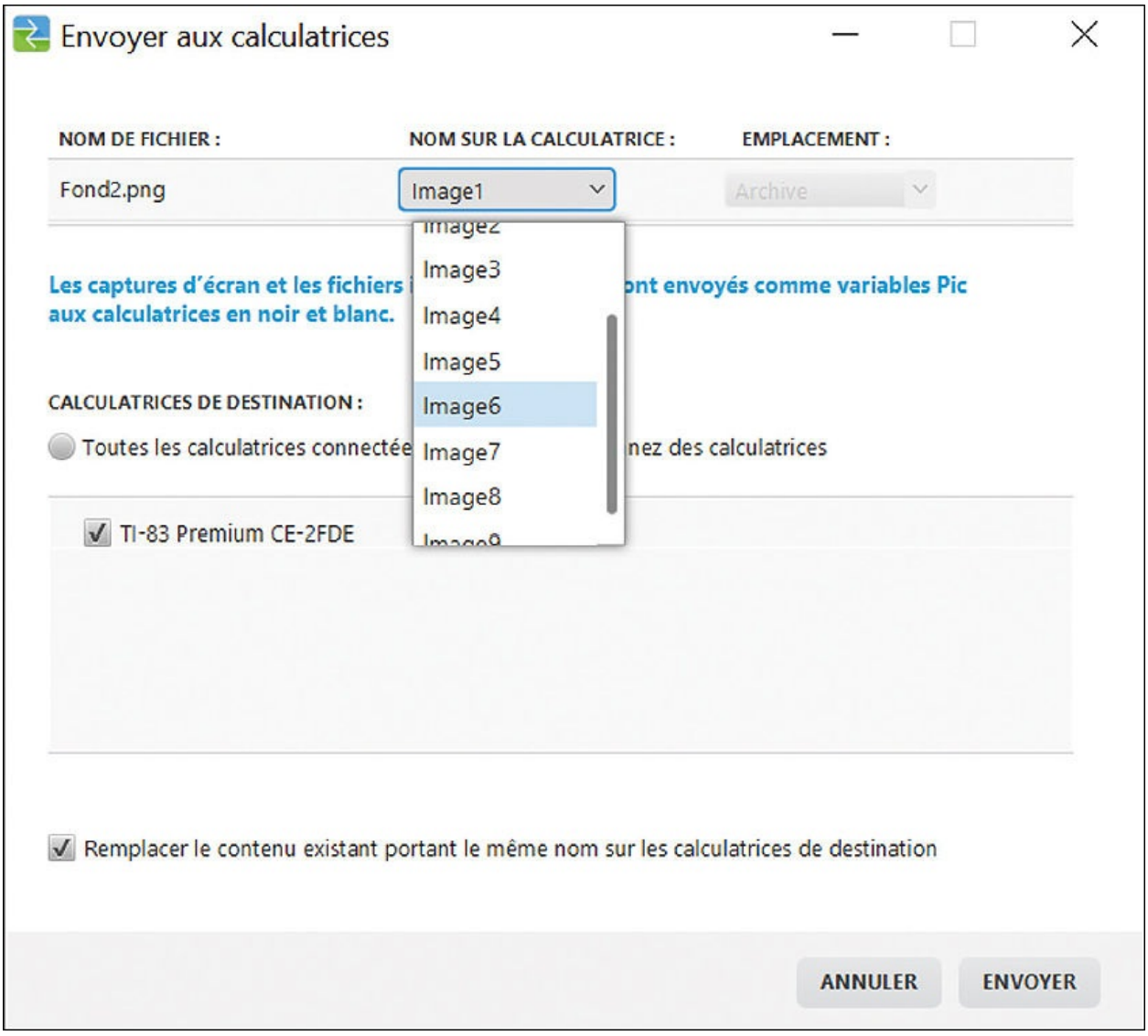
Sur votre ordinateur, ouvrez le logiciel TI Connect™ CE. Cliquez ensuite à gauche sur l'icône **`Éditeur de programmes`** et écrivez votre programme. Puis, après vous être assuré que la calculatrice est bien branchée à l'ordinateur via son câble USB, transférez le programme sur la TI-83 CE en cliquant sur le bouton . Le programme transféré aura le même nom que celui d'origine (ici, `MONTEE`).



C’est également le logiciel TI Connect™ CE qui vous permettra de transférer sur votre calculatrice les fonds d’écran de jeu que vous aurez créés sur votre ordinateur. Il convertira ces images dans un format compréhensible par la TI-83 Premium CE.

Pour transférer une image sur la calculatrice, connectez cette dernière à votre ordinateur. Dans TI Connect™ CE, cliquez sur l’icône **`Explorateur de calculatrice`**, puis sur le bouton de transfert de l’ordinateur vers l’unité nomade . Choisissez alors le fichier à transférer et son emplacement sur la calculatrice.


TI Connect™ CE
Fichier
Modifier
Afficher
Actions
Aide
CAPTURE D'ÉCRAN
EXPLORATEUR DE CALCULATRICE
ÉDITEUR DE PROGRAMMES
CALCULATRICES CONNECT... (1)
TI-83 Premium CE - 2FDE
OS 5.2
TI-83 Premium CE
Archive : 1 808 ko disponibles | RAM : 149 ko disponibles
NOM
TYPE
TAILLE
EMPLACEMENT
EasyData Application Flash 167ko Archive
Prob Sim Application Flash 71ko Archive
CelSheet Application Flash 123ko Archive
Transfrm Application Flash 25ko Archive
CabriJr Application Flash 102ko Archive
Periodic Application Flash 46ko Archive
SciTools Application Flash 54ko Archive
Conics Application Flash 42ko Archive
PlySmlt2 Application Flash 87ko Archive



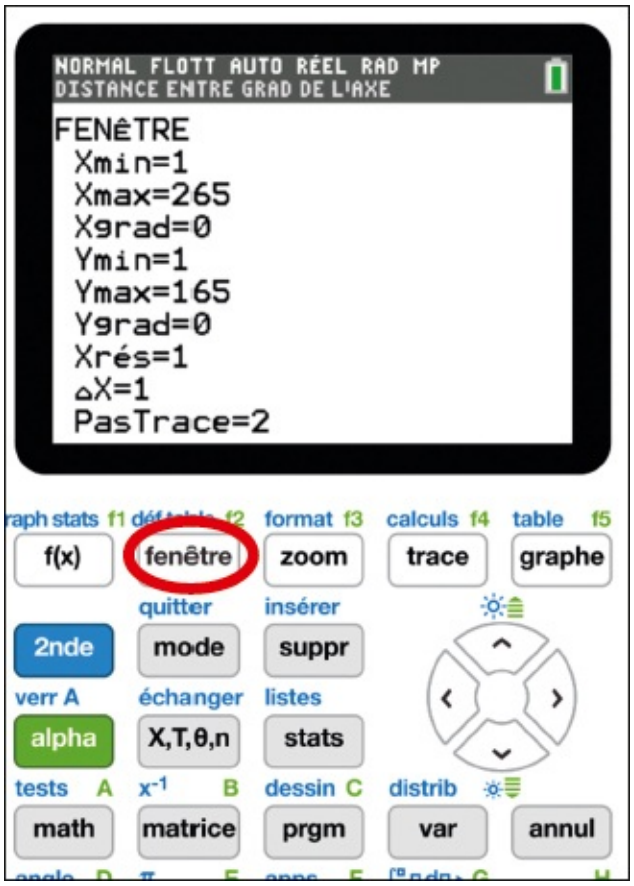
Programmation graphique

Dans les chapitres de ce livre, vous allez générer des images et des graphismes avec votre calculatrice. Mais savez-vous configurer son environnement graphique et utiliser ses fonctions de dessin ?

Fenêtrage

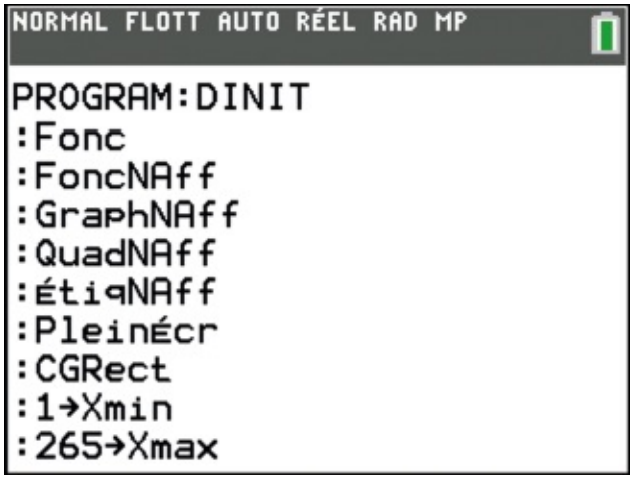
Pour bien dessiner sur la TI-83 Premium CE, il faut partir d'un écran vide aux dimensions adéquates. Mais ce n'est pas toujours le cas selon le travail qui y a précédé : présence des axes, de fonctions, etc. Vous devrez alors configurer la

fenêtre graphique de la calculatrice, soit manuellement via la touche [fenêtre], soit en définissant directement dans vos programmes les variables d'environnement graphique, auxquelles on accède par la touche [var].



Ainsi, dans l'extrait de programme ci-après, on demande à la calculatrice de se mettre en mode fonction (**Fonc**), de ne pas afficher les fonctions si certaines sont

actives (**FoncNAff**), de ne pas afficher les axes (**GraphNAff**), etc.

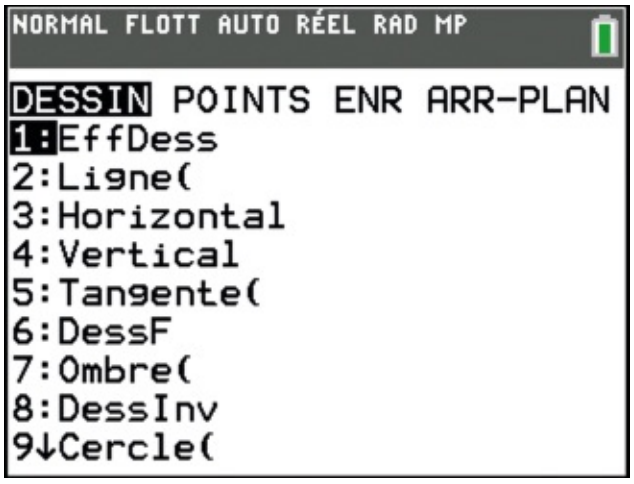


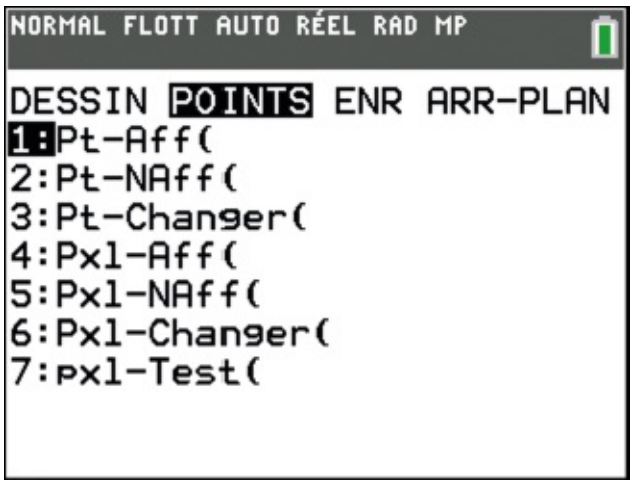
Les instructions ci-avant sont accessibles via les touches [mode], [var] et onglet VAR Y, [2nde] et [f(x)], et [2nde] et [zoom].

Primitives de dessin

Pour utiliser les fonctions graphiques de votre calculatrice dans vos programmes, appuyez sur les touches [2nde], [prgm], puis naviguez dans les onglets **DESSIN, POINTS, ENR** et **ARR-PLAN**.

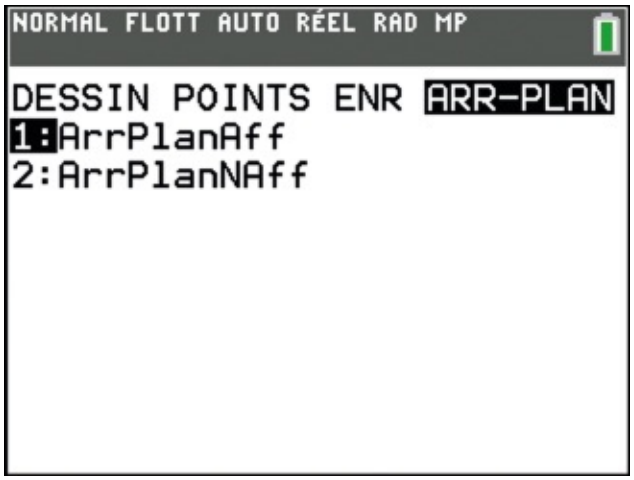
C'est dans l'onglet **DESSIN** que vous trouverez par exemple la routine de tracé `Ligne`, et dans l'onglet **POINTS** les routines relatives aux manipulations de pixels.



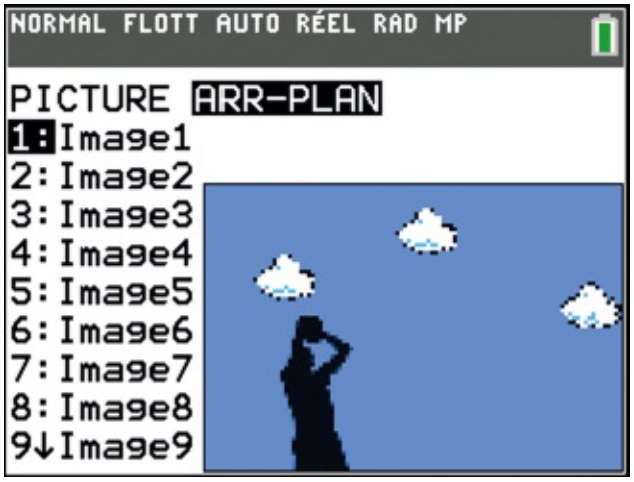


Savez-vous qu'en positionnant le curseur en regard d'une fonction à l'aide des flèches et qu'en pressant la touche [+], vous faites apparaître un menu contextuel d'aide ? Bien pratique pour retrouver les paramètres d'une fonction !

Attention, ne confondez pas l'onglet **`ARR-PLAN`**, accessible par les touches [2nde], [prgm] (dessin C), avec le menu **`Pic et arrière-plan`**, accessible par la touche [var] et l'onglet **`Variables`**. Le premier donne accès à la fonction **`ArrPlanAff`**, qui permet dans un programme d'aller chercher un arrière-plan pour l'afficher :



Le second permet de visualiser les images d'arrière-plan disponibles dans votre calculatrice, préalablement transférées à l'aide de TI Connect™ CE.



Programmation avec le TI-Innovator™ Hub

Dans ce livre, nous allons découvrir le très grand potentiel du TI-Innovator™ Hub, un petit boîtier qui se branche sur la calculatrice TI-83 Premium CE. Grâce à ses composants intégrés (microcontrôleur, LED, capteur de lumière, haut-parleurs…) et ses différents ports d'entrée/sortie, il permet de développer de multiples projets. Regardons ici comment le faire fonctionner, puis testons-le avec quelques opérations simples.

Installons l'application TI-Innovator™ Hub

Pour pouvoir utiliser le TI-Innovator™ Hub, vous devez d'abord vérifier que votre TI-83 Premium CE utilise bien le système d'exploitation 5.2 ou ultérieur : appuyez sur [2nde], [mém +], et sélectionnez **`À propos de`**. Si ce n'est pas le cas, mettez à jour votre calculatrice via **`https://education.ti.com/fr/product-resources/whats-new-84-ce`**.

Pour vérifier si l'application TI-Innovator™ Hub est déjà installée sur votre calculatrice, appuyez sur [2nde], [mém +], sélectionnez `Gest. Mémoire/Suppr.` puis `Apps`, et recherchez `Hub` dans la liste des applications installées.

Vous devez ensuite télécharger et installer l'application TI-Innovator™ Hub sur

votre calcultatrice, sauf si elle y est déjà. Voici comment procéder.

1. Assurez-vous d'avoir installé la version 5.2 ou ultérieure du logiciel TI Connect™ CE sur votre ordinateur (voir page 9).

2. Sur votre ordinateur, allez à l'adresse **`https://education.ti.com/fr/software/details/en/150E0DDCE2FF46C0BB96C5230970F967/ti-innovator-hub`**.

3. Téléchargez l'application TI-Innovator™ Hub. Notez les noms du fichier téléchargé et du dossier dans lequel vous l'enregistrez.

4. Connectez votre calculatrice à l'ordinateur à l'aide du câble USB fourni.

5. Sur l'ordinateur, ouvrez le logiciel TI Connect™ CE, qui doit reconnaître la calculatrice.

6. Dans le menu **`Actions`**, sélectionnez **`Ajouter des fichiers depuis un ordinateur`**. TI Connect™ CE vous invite alors à choisir un fichier.

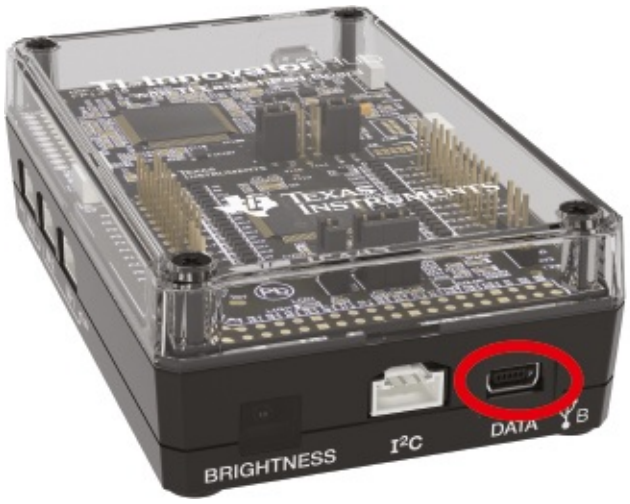
7. Naviguez jusqu'au dossier où vous avez enregistré le fichier téléchargé et double-cliquez sur ce dernier.

8. Dans la fenêtre **`Envoyer aux calculatrices`** qui apparaît, cliquez sur **`ENVOYER`**.

Votre TI-Innovator™ Hub est maintenant prêt à l'emploi ! Si vous allez dans l'éditeur de programmes, vous constaterez qu'en appuyant sur la touche prgm, un onglet **`HUB`** est apparu en haut, preuve que l'application TI-Innovator Hub™ est bien installée.

Allumons une LED

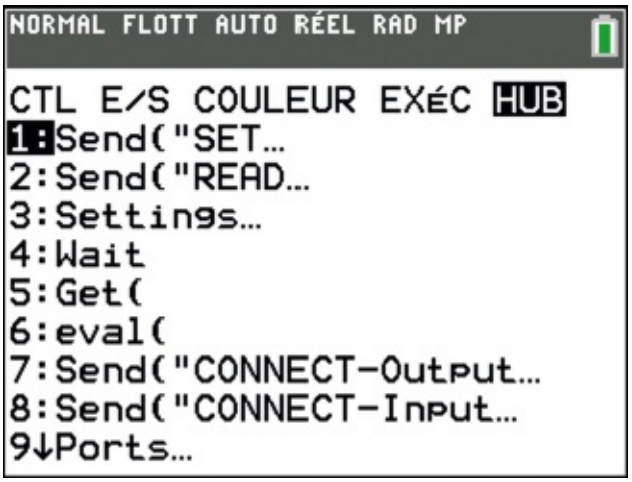
À l'aide du câble USB fourni, reliez le TI-Innovator™ Hub à la TI-83 Premium CE, en branchant le connecteur USB de type B sur le Hub, comme signalé par le pictogramme. Le connecteur de type A doit être relié à la calculatrice.

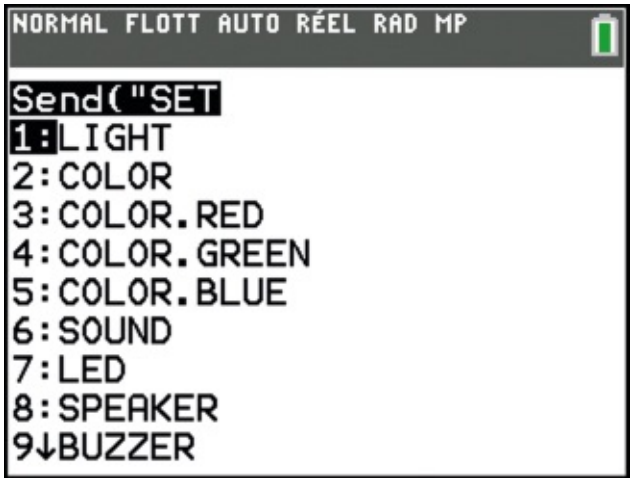


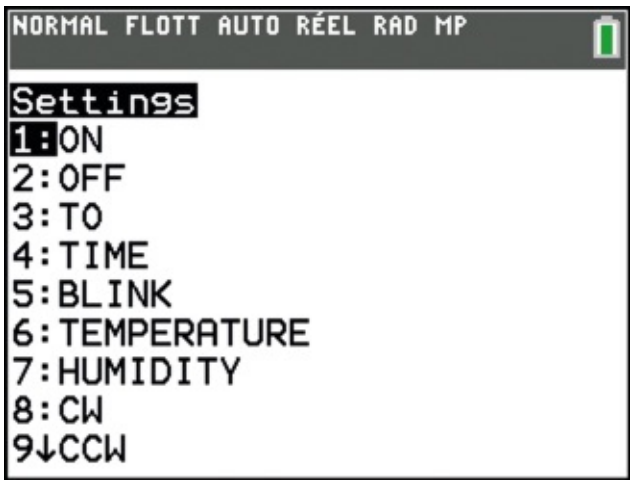
Créez un programme **`LIGHT`** et saisissez les lignes de code suivantes en naviguant dans les menus **`Send(«SET`** et **`Settings`** de l'onglet **`HUB`** :

```
Send("SET LIGHT ON")
Wait 1
Send("SET LIGHT OFF")
Wait 1
```

Lorsque vous éditez un programme nécessitant l'utilisation du TI-Innovator™ Hub, vous trouverez les routines permettant de le piloter dans l'onglet HUB, accessible par la touche [prgm].







Attention ! Rappelez-vous que la touche [prgm] ne donne pas accès aux mêmes menus selon qu'on est dans l'écran de calcul ou dans l'éditeur de programmes.

Si on exécute le programme, la diode rouge du TI- Innovator™ Hub s'allume puis s'éteint au bout d'une seconde.

Pour exécuter un programme à partir de l'écran de calcul, appuyez sur la touche [prgm] et sélectionnez-le dans la liste. Validez une première fois via [entrer] pour que le nom du programme apparaisse dans l'écran de calcul, puis validez à nouveau pour l'exécuter.

Ouvrons le TI-Innovator™ Hub à son environnement

Grâce à ses multiples ports d'entrée/sortie, le TI-Innovator™ Hub a la possibilité de communiquer avec son environnement. Il est ainsi compatible avec la plupart des capteurs et actionneurs Grove, qui sont disponibles en kit ou au détail chez les revendeurs de fournitures électroniques.

Commençons par tester un capteur à ultrasons, en le branchant sur le port IN 1 du Hub à l'aide du câble normalisé fourni.

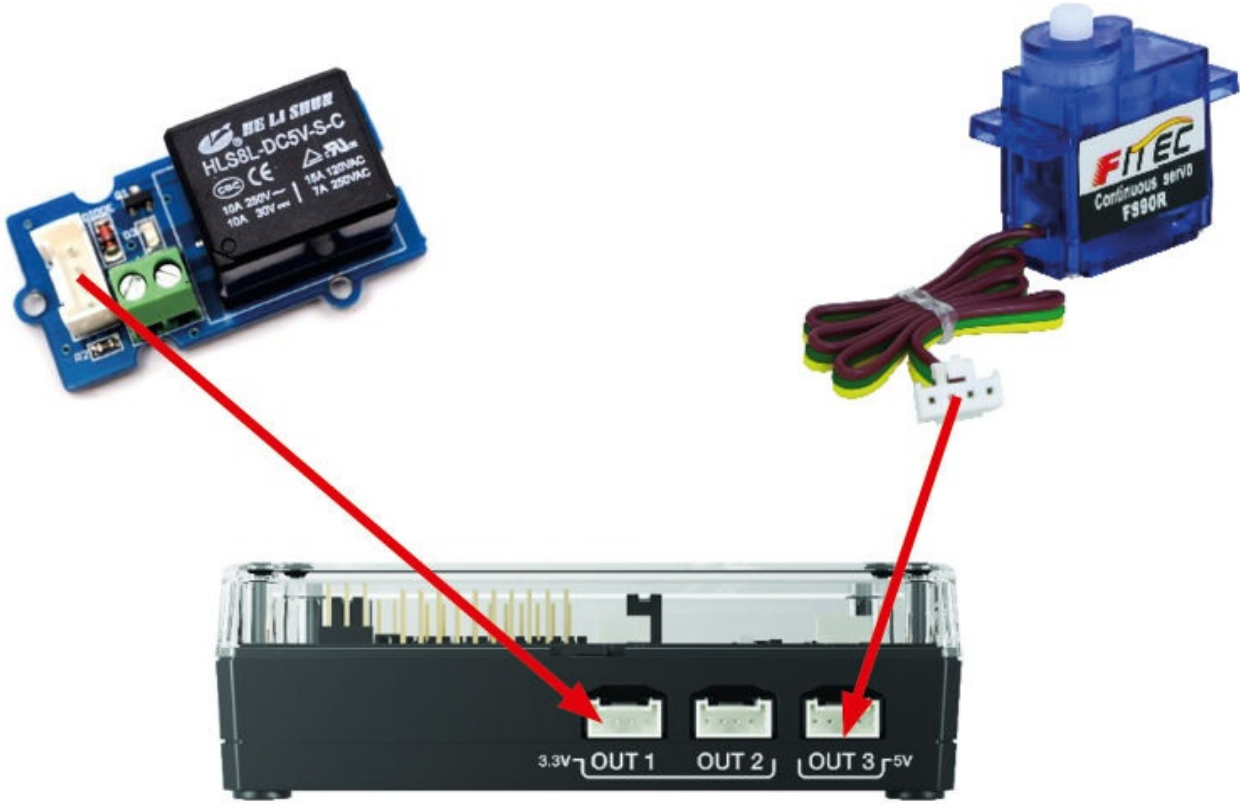
Attention, certains capteurs doivent être branchés sur le port IN 3 du Hub en raison de leur tension d'utilisation.

Saisissez le programme suivant :

```
Send(“CONNECT RANGER 1 TO IN1”)
Send(“READ RANGER 1”)
Get(R)
Disp(R)
```

Lors de son exécution, ce programme renverra la valeur de la distance séparant le capteur à ultrasons de l’obstacle le plus proche.

Branchons à présent des actionneurs Grove (relais et servomoteur) sur les ports OUT du TI-Innovator™ Hub à l’aide des câbles fournis. On fera attention à leur tension d’utilisation : le relais sera branché sur le port OUT 1, mais le servomoteur devra être connecté au port OUT 3. Il faut en outre ajouter une alimentation externe en la branchant sur le port microUSB PWR.



Exercice

Déterminer ce qu'il va se passer avec le programme suivant.

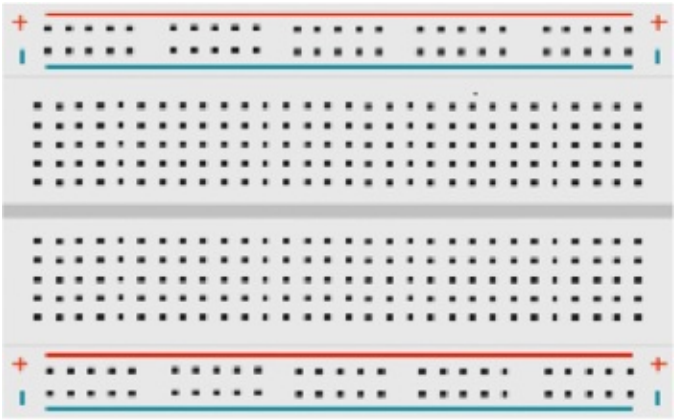
```
Send("CONNECT SERVO 1 TO OUT 3")
Send(«SET SERVO 1 TO -90»)
```

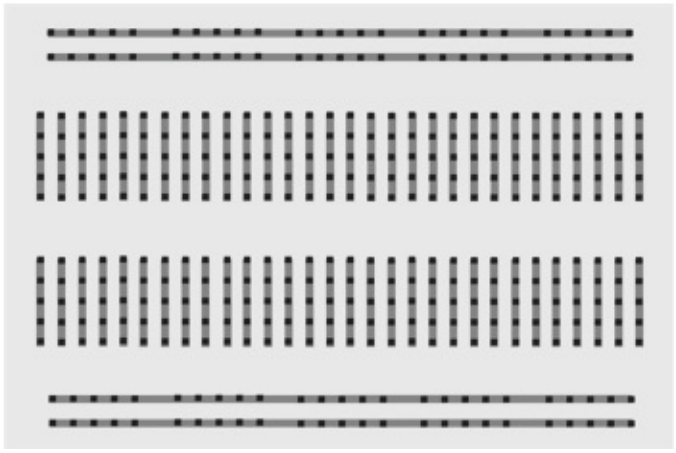
http://go.eyrolles.com/ti-intro

Étudions maintenant un autre mode de communication du TI-Innovator™ Hub avec son environnement : la platine d'expérimentation (*breadboard* en anglais). Appelée encore platine d'essai, cette plaque permet de réaliser des montages électroniques sans soudure, grâce à ses broches qui sont reliées entre elles par des bandes métalliques.

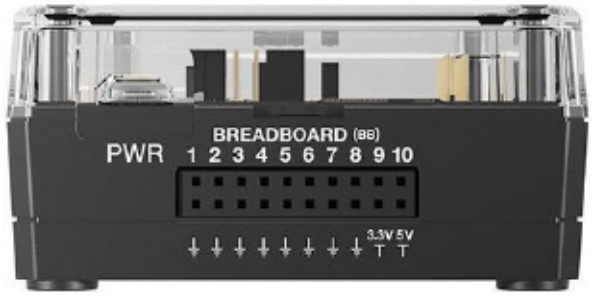
La plupart des broches sont regroupées par colonnes de cinq. Toutes les broches d'une même colonne sont connectées électriquement les unes aux autres, mais chaque colonne est isolée des autres. De part et d'autre de ces colonnes se trouvent des rails d'alimentation, signalés par des bandes rouges (+) et bleues (–). Les broches d'un rail sont reliées entre elles sur toute la longueur de la bande.

Voici une platine vue de dessus (image du haut) et de dessous (image du bas).





Quant aux broches du Hub, elles sont au nombre de 20 au total : 10 broches de signal, 8 broches de mise à la terre, une broche d'alimentation de 3,3 V et une broche d'alimentation de 5 V.

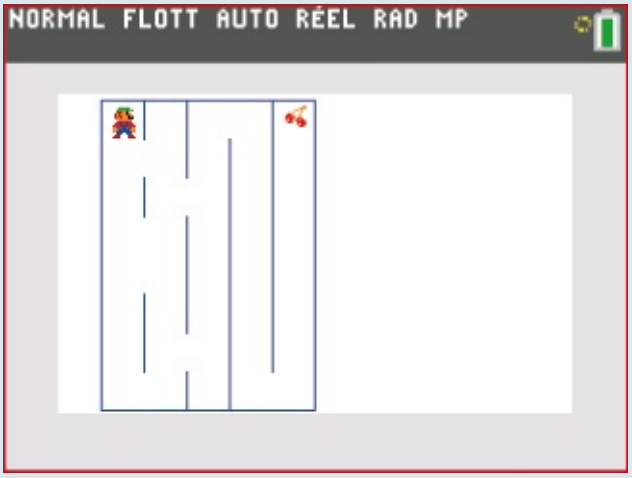


Nous utiliserons une platine d'expérimentation dans le dernier chapitre de ce livre.

Chapitre 1

Le labyrinthe

Le but de ce chapitre est de créer un labyrinthe dans lequel il faut déplacer un personnage jusqu'à la sortie à l'aide du clavier.



ÉTAPE 1 **Savoir sur quelles touches du clavier appuie le joueur afin de programmer le mouvement du personnage.**

ÉTAPE 2 **Apprendre à afficher un point à l'écran.**

ÉTAPE 3 **Dessiner le personnage puis l'afficher sur l'écran en utilisant les pixels de couleur.**

ÉTAPE 4 **Créer le labyrinthe et gérer le mouvement du personnage à l'intérieur.**

ÉTAPE 1 Les codes des touches

Pour déplacer le personnage dans le labyrinthe, le joueur devra appuyer sur les flèches de direction. Il faut donc savoir quelles touches du clavier il presse pour déterminer le mouvement du personnage.

À chaque touche du clavier est associé un nombre. Pour savoir dans un programme quelle touche est enfoncée, il faut utiliser l'instruction **GetKey**, accessible via la touche prgm, onglet **E/S**. Cette instruction renvoie le code de la touche qui a été appuyée, ou 0 si aucune touche n'est pressée. Mais attention : dès qu'on a fini d'appuyer sur une touche, **GetKey** reprend la valeur 0. C'est une fonction dont la valeur est actualisée à chaque millième de seconde.

Il faut donc écrire un programme qui teste constamment la valeur de **GetKey** (qui vaut 0 tant qu'une touche n'est pas enfoncée), et qui s'arrête dès qu'une touche a été pressée, en affichant le code de la touche. Ce programme, qui affiche le code de la touche sur laquelle on appuie, est présenté ci-après.

```
0→B
While B=0
getKey→A
If A≠0
Then
A→B
EffÉcran
Disp A
End
End
```

Cette boucle va s'exécuter tant que B=0.

DÉFI 1

Comment modifier le programme ci-avant pour que, sans être relancé, il affiche le code de chaque touche enfoncée ? Indication : il suffit de supprimer une ligne…

Exercice 1

Exécuter le programme précédent et déterminer le code des touches ◀,

, et .

Pour exécuter un programme, il faut :

1. sortir de l'éditeur de programmes via 2nde, quitter mode ;
2. choisir le programme à exécuter via prgm, onglet EXEC ;
3. sélectionner le nom du programme souhaité.

go.eyrolles.com/ti-ch1ex1

Exercice 2

Sur cette calculatrice, inscrire le code de chacune des touches.

go.eyrolles.com/ti-ch1ex2

Exercice 3

Compléter les pointillés du programme suivant pour qu'il affiche à l'écran un compteur c qui démarre à 5 et qui augmente de 10 chaque fois qu'on appuie

sur la touche ▶.

```
5→C
0→B
While B=0
getKey→A
If A=26
Then
……→C
Output(10,14,C)
End
End
```

La boucle ci-avant va s'exécuter tant que `B=0`. Comme `B` ne changera pas de valeur, le programme ne s'arrêtera jamais !

`Output` permet d'afficher un texte ou une variable à un endroit précis de l'écran, comme ici ligne 10 et colonne 14.

go.eyrolles.com/ti-ch1ex3

Exercice 4

Modifier le programme précédent pour que C diminue de 10 lorsqu'on appuie sur la touche ◀.

go.eyrolles.com/ti-ch1ex4

Exercice 5

Modifier le programme précédent pour afficher, à gauche de l'écran, la variable `c` (dont la valeur change avec les flèches ◄ et ►) et, à droite de l'écran, une variable `L` qui varie avec les flèches ▲ et ▼.

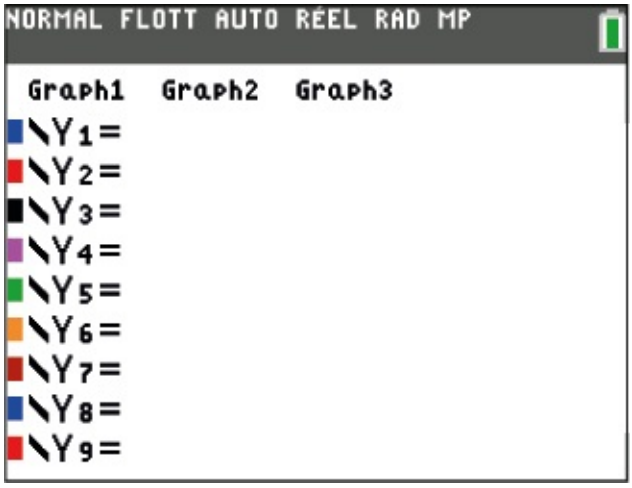
`go.eyrolles.com/ti-ch1ex5`

ÉTAPE 2 Affichage d'un point à l'écran

Pour afficher un point à l'écran, il existe deux méthodes, selon qu'on se réfère aux coordonnées du repère ou à celles des pixels.

Mais tout d'abord, il faut supprimer à l'écran tout affichage non souhaité (tracé de fonctions, graphique statistique, etc.). Vous devez donc effacer les fonctions présentes dans [f(x)] et désactiver **`Graph1`**, **`Graph2`** et **`Graph3`** qui ne doivent pas être en surbrillance (si vous voyez par exemple **`Graph1`**, c'est qu'il est activé !).

Votre écran doit ressembler à celui ci-après.

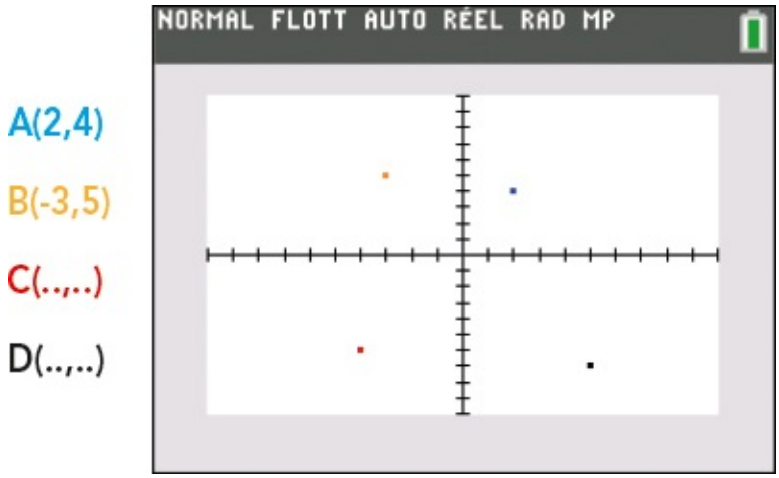


Pour désactiver un `Graph`, se positionner dessus avec les flèches de direction et appuyer sur [entrer].

En utilisant les coordonnées du repère

Exercice 6

Déterminer les coordonnées des points C et D :

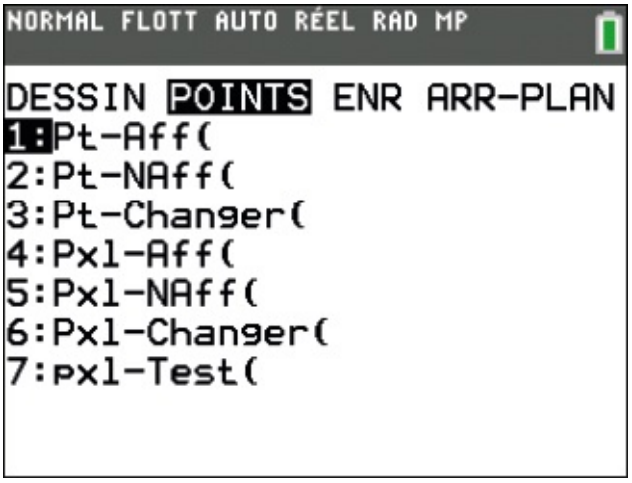


`go.eyrolles.com/ti-ch1ex6`

Le repère peut être modifié en appuyant sur [fenêtre].

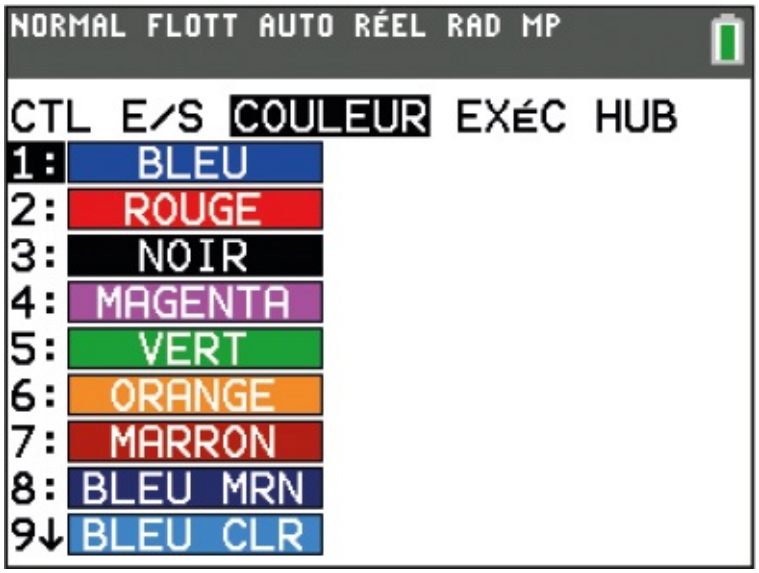
Si on utilise les coordonnées du repère, l'instruction permettant d'afficher un point à l'écran est alors **`Pt-Aff(abscisse,ordonnée)`**. Elle est accessible via [2nde], [dessin C prgm], onglet **`POINTS`**.

Pour afficher le point A en bleu, il faut écrire la ligne de code **`Pt-Aff(2,4,BLEU)`** en récupérant le **`BLEU`** dans l'onglet **`COULEUR`** via la touche [prgm].



Exercice 7

Écrire un programme qui permet d'afficher les points précédents A, B, C et D en respectant leur couleur.



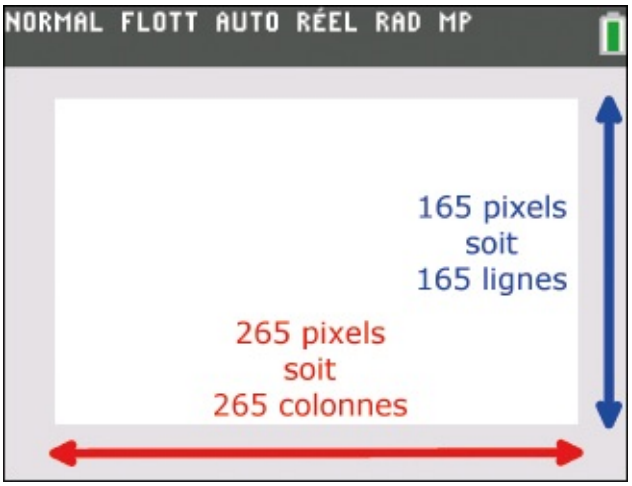
On accède aux noms des couleurs disponibles via prgm, onglet COULEUR.

go.eyrolles.com/ti-ch1ex7

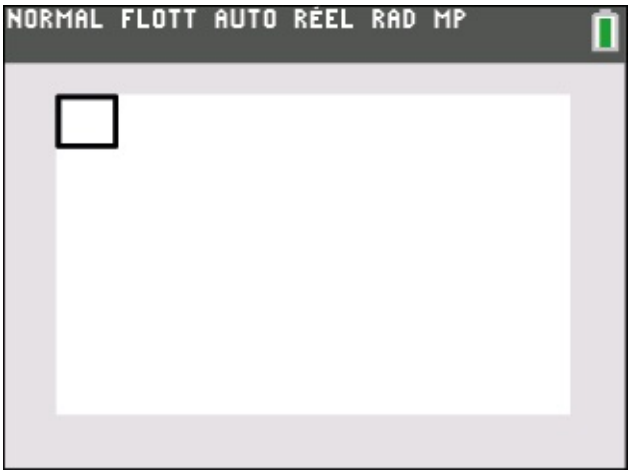
En utilisant les coordonnées des pixels

Si on n'utilise pas le repère précédent, il faut d'abord supprimer son affichage en choisissant Axes : NAFF via 2nde, zoom.

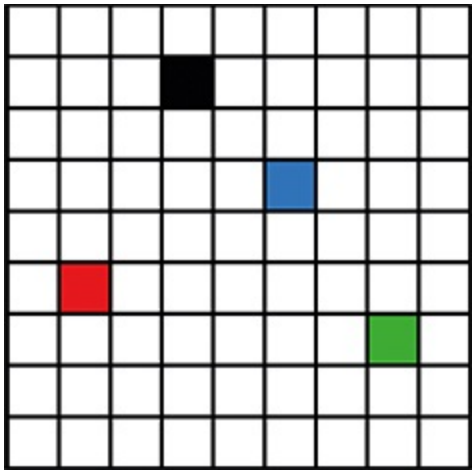
Pour repérer un point sur l'écran de la TI-83 Premium CE, on peut aussi se référer aux coordonnées des pixels, sachant que cet écran a une largeur de 265 pixels et une hauteur de 165 pixels. Dans ce cas, l'instruction pour afficher un point est **Pxl-Aff** (ligne,colonne,**[couleur]**), accessible via 2nde, prgm.



Attention ! On ne se repère plus avec le repère habituel… mais par ligne et par colonne (l'origine du nouveau repère étant en haut à gauche). Pour bien comprendre ce principe, effectuons un zoom du coin supérieur gauche de l'écran :



et représentons la zone agrandie ci-après en activant quatre pixels de couleurs différentes.



Le pixel noir est situé ligne 2, colonne 4, donc l'instruction pour l'afficher est :
`Pxl-Aff(2,4,NOIR)`

Dans un programme qui dessine (des pixels ou autres), il faut prévoir au début l'instruction `EffDess` (accessible via [2nde], [dessin C prgm]) pour effacer les anciens dessins.

Exercice 8

Donner les instructions pour afficher les trois autres pixels de couleur.

`go.eyrolles.com/ti-ch1ex8`

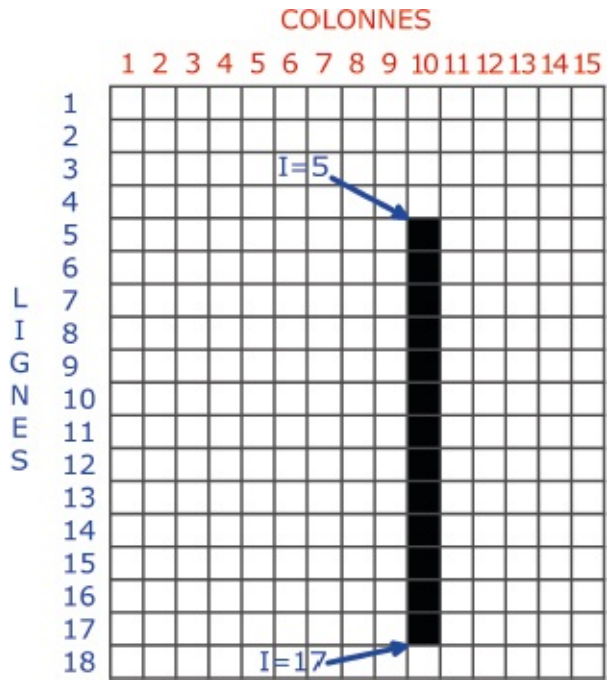
Exercice 9

Écrire un programme qui affiche les quatre pixels précédents (ouvrez bien les yeux, c'est tout petit !).

Le dessin pixel par pixel peut s'avérer très long ! Représentons un segment avec la boucle **FOR** grâce à ce programme nommé **CARREMVT** :

```
EffDess
For(I,5,17)
Pxl-Aff(I,10,NOIR)
End
```

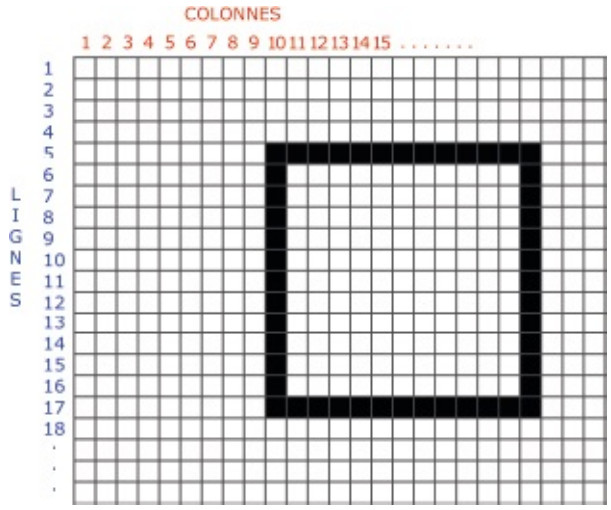
À l'exécution, on peut voir le segment ci-après qui s'affiche.



go.eyrolles.com/ti-ch1ex9

Exercice 10

Compléter le programme précédent afin de représenter le carré ci-dessous :



go.eyrolles.com/ti-ch1ex10

DÉFI 2

Comment écrire un programme en cinq lignes qui colorie tout l'écran en bleu ?

ÉTAPE 3 Affichage et déplacement du personnage

Nous allons maintenant déplacer le carré précédent à l'aide des flèches de direction. Pour commencer, écrivons un programme qui le fait avancer de 10 pixels à droite lorsqu'on appuie sur la touche (touche qui a pour code 26).

En réalité, nous avons besoin de deux programmes.

- Le programme **CARRE** qui permet d'afficher le carré :

```
For(I,5,17)
```

- Le programme **PRINCIPA** qui fait augmenter de 10 une variable **C** (démarrant à 5) lorsqu'on appuie sur

```
Pxl-Aff(I,10,NOIR)
Pxl-Aff(I,22,NOIR)
End

For(I,10,22)
Pxl-Aff(5,I,NOIR)
Pxl-Aff(17,I,NOIR)
End
```

la flèche ▶ :

```
DispGraph
EffDess
5→C
0→B
While B=0
getKey→A
If A=26
Then
C+10→C
End
End
```

L'instruction `DispGraph`, accessible via prgm, onglet `E/S`, permet d'afficher la fenêtre graphique.

Pour faire le lien entre ces deux programmes, nous allons d'abord modifier les instructions **`Pxl-Aff`** afin de tenir compte de **`C`**. En particulier, les coordonnées du carré doivent dépendre de cette variable. De plus, dans le programme **`PRINCIPA`**, dès que **`C`** change de valeur, il faut demander d'exécuter le dessin du carré, c'est-à-dire le programme **`CARRE`**. On aboutit donc aux programmes suivants.

- Programme **`CARRE`** :

```
For(I,5,17)
Pxl-Aff(I,C+10,NOIR)
Pxl-Aff(I,C+22,NOIR)
End

For(I,10,22)
Pxl-Aff(5,C+I,NOIR)
Pxl-Aff(17,C+I,NOIR)
End
```

- Programme **`PRINCIPA`** :

```
DispGraph
EffDess
5→C
0→B
While B=0
getKey→A
If A=26
Then
C+10→C
prgmCARRE
End
End
```

Pour écrire l'instruction `prgmCARRE`, appuyer sur prgm,

onglet **EXEC** et choisir **CARRE.**

Si on exécute le programme **PRINCIPA** et qu'on appuie sur la flèche , le carré se déplace ! Mais catastrophe, on constate que le carré précédent reste visible ☹. Il faut donc l'effacer…

Pour cela, on va utiliser l'instruction **Pxl-NAff(ligne, colonne)**. Accessible via , onglet **POINTS**, elle permet d'éteindre un pixel en le coloriant de la même couleur que le fond d'écran (donc en blanc ici) avant de dessiner notre nouveau carré.

Pour connaître la position précédente du carré, nous allons introduire une variable **E** qui correspondra à l'ancienne valeur de **C**. **E** sera donc définie juste avant le changement de valeur de **C**.

- Programme **CARRE** :

```
For(I,5,17)
Pxl-NAff(I,E+10)
Pxl-NAff(I,E+22)
End

For(I,10,22)
Pxl-NAff(5,E+I)
Pxl-NAff(17,E+I)
End

For(I,5,17)
Pxl-Aff(I,C+10,NOIR)
Pxl-Aff(I,C+22,NOIR)
End

For(I,10,22)
Pxl-Aff(5,C+I,NOIR)
Pxl-Aff(17,C+I,NOIR)
End
```

- Programme **PRINCIPA** :

```
DispGraph
EffDess
5→C
0→B
While B=0
getKey→A
If A=26
Then
C→E
C+10→C
prgmCARRE
End
End
```

Exercice 11

Modifier le programme PRINCIPA pour que le carré puisse se déplacer vers la gauche.

go.eyrolles.com/ti-ch1ex11

Exercice 12

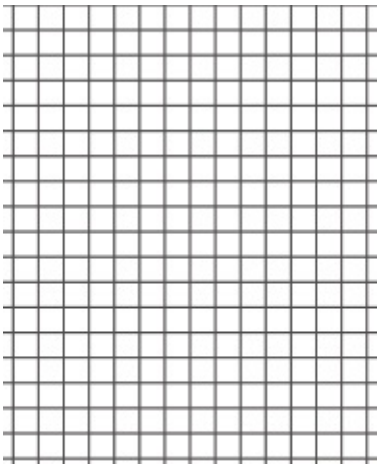
Modifier les programmes `CARRE` et `PRINCIPA` pour pouvoir déplacer le carré à l'aide des flèches ▲ et ▼. On pourra introduire une variable `L` (pour ligne…).

go.eyrolles.com/ti-ch1ex12

À présent, il faut dessiner le personnage ! Si vous souhaitez représenter l'image , il faut la dessiner point par point…

Voici notre personnage pixel par pixel :

Dessiner ci-dessous son propre personnage :



Dans le programme qui va dessiner le personnage, il est possible d'utiliser quelques boucles pour simplifier l'écriture du code. En effet, la ligne 1 ne contient que des pixels verts. Cette ligne peut donc s'écrire :

```
For(I,5,10)
Pxl-Aff(L+2,C+I,vert)
End
```

Ce qui est plus court que :

```
Pxl-Aff(L+2,C+5,vert)
Pxl-Aff(L+2,C+6,vert)
Pxl-Aff(L+2,C+7,vert)
Pxl-Aff(L+2,C+8,vert)
Pxl-Aff(L+2,C+9,vert)
Pxl-Aff(L+2,C+10,vert)
```

Pour gérer le mouvement du personnage, nous avons introduit les variables `L` (pour ligne) et `C` (pour colonne).

On procédera de la même manière pour la deuxième ligne. En revanche, pour la ligne 3, on devra écrire :

```
Pxl-Aff(L+4,C+4,ROUGE)
Pxl-Aff(L+4,C+5,ROUGE)
Pxl-Aff(L+4,C+6,ROUGE)
Pxl-Aff(L+4,C+7,ORANGE)
Pxl-Aff(L+4,C+8,ORANGE)
Pxl-Aff(L+4,C+9,NOIR)
Pxl-Aff(L+4,C+10,ORANGE)
```

Exercice 13

Écrire le programme `PERS` qui affiche le personnage. Puis animer ce dernier en exécutant et en modifiant le programme `PRINCIPA`.

On remarque qu'à chaque déplacement, le personnage reste affiché. Il faut donc écrire un programme qui va effacer l'ancien personnage avant d'afficher le nouveau. On pourra utiliser l'instruction **`Pxl-NAff`**.

`go.eyrolles.com/ti-ch1ex13`

Exercice 14

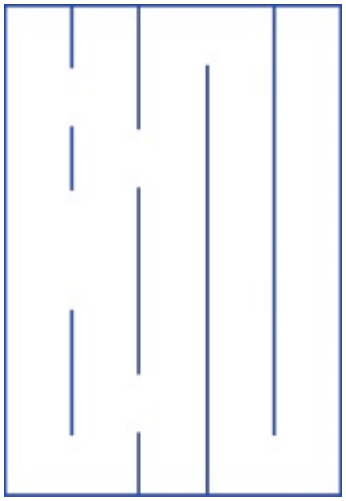
Écrire le programme `PERSOFF` qui efface le personnage, puis modifier le programme `PRINCIPA` en conséquence. Dans ce dernier, il faudra appeler le programme `PERSOFF` en ayant préalablement stocké les coordonnées de l'ancienne position du personnage.

`go.eyrolles.com/ti-ch1ex14`

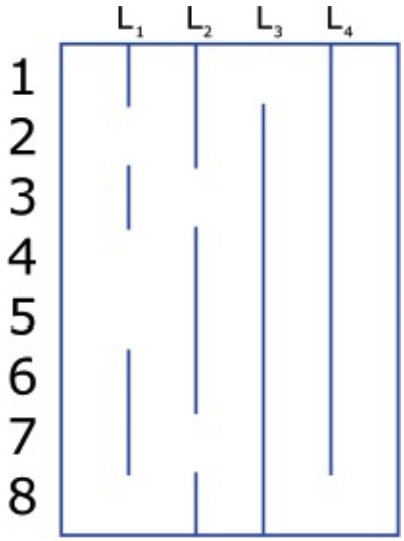
ÉTAPE 4 Construction du labyrinthe et finalisation du jeu

Dans la programmation d'un labyrinthe, la difficulté est de traduire en code dans quel cas le personnage a le droit d'effectuer un déplacement. Comment allons-nous créer l'objet informatique labyrinthe ?

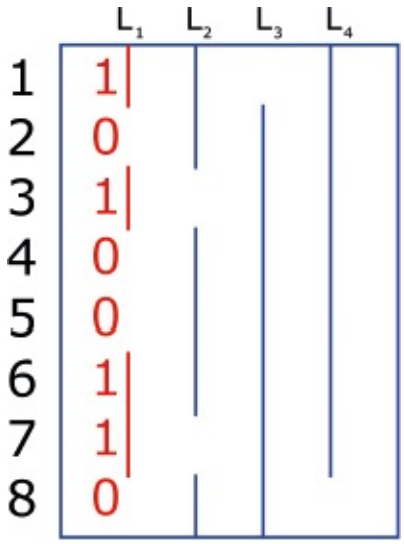
Partons du labyrinthe ci-après :



Modélisons les murs en considérant que les quatre lignes bleues discontinues **L**1 à **L4** contiennent parfois des murs et parfois non. Sur chacune de ces lignes verticales, la présence d'un mur sera codée par un 1, et son absence par un 0.



Avec ce système, la première ligne du labyrinthe sera codée de la manière ci-après :



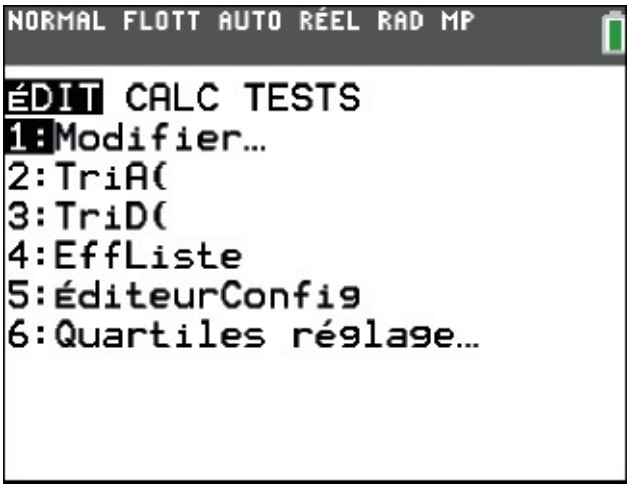
Exercice 15

Poursuivez le codage des lignes L_2, L_3 et L_4.

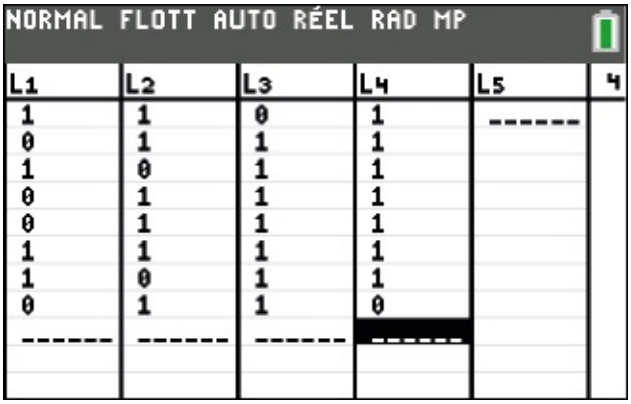
Précisons que vous n'êtes pas obligé de prendre comme modèle le labyrinthe ci-dessus. Vous pouvez dessiner vos propres murs et les coder !

go.eyrolles.com/ti-ch1ex15

Nous allons maintenant entrer les codes correspondant aux murs dans des listes, en appuyant sur stats et en choisissant **Modifier**. Cela nous permettra de lire ces valeurs lors de la construction du labyrinthe, ou éventuellement de les modifier si on souhaite changer son aspect.



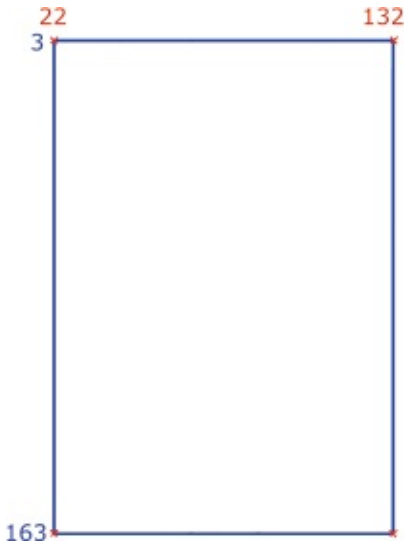
Entrons alors les valeurs déterminées précédemment :



S'il manque des listes, appuyer sur stats**, puis choisir** `ÉditeurConfig`**.**

Pour effacer les listes, appuyer sur stats**, puis entrer** `EffListe` L_1, L_2, L_3, L_4.

Il faut maintenant écrire le programme **LABY** qui va dessiner le labyrinthe en tenant compte des murs (qu'on pourra éventuellement changer par la suite en modifiant les valeurs des listes L_1, L_2, L_3 et L_4 correspondant aux lignes du même nom). Commençons par en dessiner le cadre ci-après.



On peut utiliser une boucle pour afficher les deux lignes verticales :

```
EffDess
For(I,22,132)
Pxl-Aff(3,I,BLEU)
Pxl-Aff(163,I,BLEU)
End
```

Exercice 16

Écrire la boucle permettant d'afficher les deux lignes horizontales du cadre.

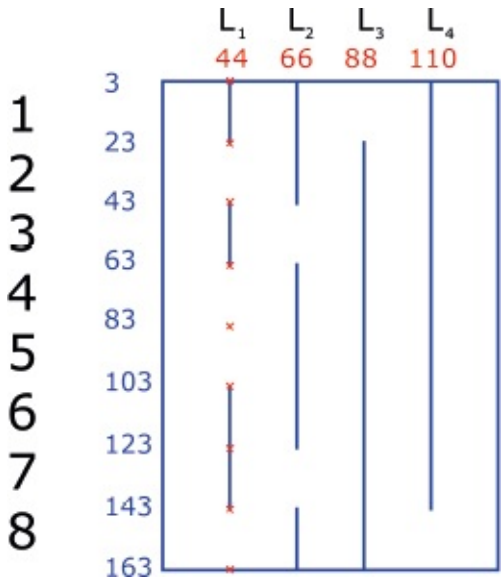
Occupons-nous à présent des murs. Nous allons créer quatre boucles, puisque ces murs ont été codés dans quatre listes. Un mur étant un segment, il faut d'abord déterminer les coordonnées des pixels des extrémités de chacun d'eux.

Pour la liste L_1, nous avons comme coordonnées (ligne, colonne) : (3,44) ; (23,44) ; (43,44) ; (63,44) ; (83,44) ; (103,44) ; (123,44) ; (143,44) et (163,44).

- Si $L_1(1)=1$, alors il faut tracer la ligne de (3,44) à (23,44), sinon on n'affiche rien.
- Si $L_1(2)=1$, alors il faut tracer la ligne de (23,44) à (43,44), sinon on n'affiche rien.
- Si $L_1(3)=1$, alors il faut tracer la ligne de (43,44) à (63,44), sinon on n'affiche rien.

  ……

- Si **L_1(8)=1**, alors il faut tracer la ligne de (143,44) à (163,44), sinon on n'affiche rien.



La liste L_1 sera donc codée de la manière suivante (pour l'instruction **Pxl-Aff**, pas besoin ici de préciser la couleur : ce sera alors par défaut la dernière utilisée par la calculatrice) :

```
For(I,1,8)
If L1(I)=1
Then
For(J,0,20)
Pxl-Aff(3+20*(I-1)+J,22*2)
End
End
End
```

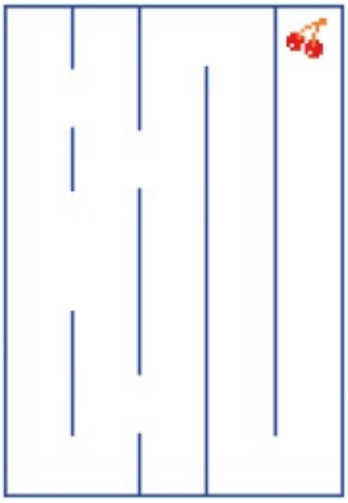
On arrive à la formule ci-avant en remarquant que les nombres 3, 23, 43,..., 143 peuvent s'écrire sous la forme 3 + 20 × (I - 1) avec I variant de 1 à 8.

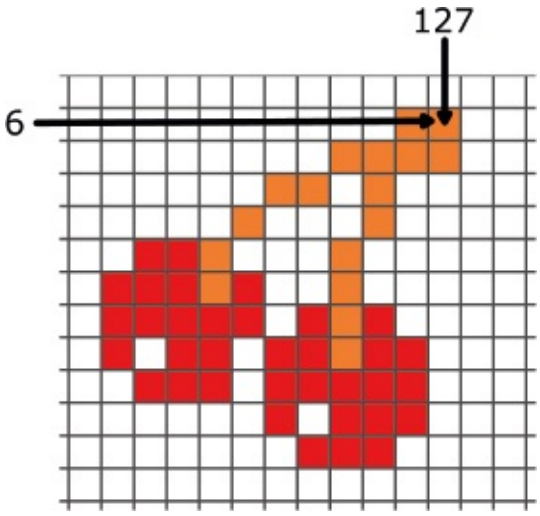
go.eyrolles.com/ti-ch1ex16

Exercice 17

Écrire le code pour les trois autres listes afin d'afficher tous les murs.

Pour marquer la fin du jeu, il faut que votre personnage atteigne la cerise. Il faut donc coder le dessin de ce fruit (ou d'un autre de votre choix). On a indiqué ici les coordonnées du pixel **`ORANGE`** de référence.





go.eyrolles.com/ti-ch1ex17

Exercice 18

Coder la cerise (ou tout autre fruit de dimension équivalente).

On a presque fini ! Il ne reste plus qu'à écrire le programme principal, appelé **MAITRE**, qui va :

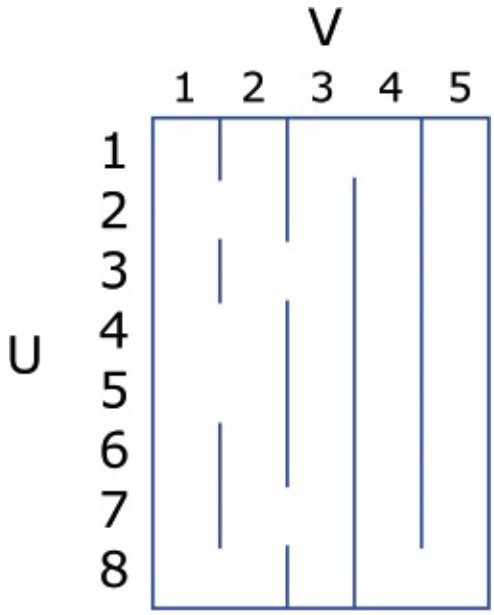
- afficher le labyrinthe ;
- afficher le personnage à sa position initiale ;
- permettre le mouvement du personnage (selon la présence ou l'absence d'un mur) ;
- s'arrêter et afficher « Gagné » lorsque le personnage aura atteint la cerise.

Le premier point est déjà réglé puisqu'il suffit d'appeler le programme **LABY** pour afficher le labyrinthe et la cerise : **prgmLABY**

Le second point est facile : il faut appeler le programme **PERS** en donnant des valeurs de départ à ses variables **L** et **C**. On prendra **L=5** et **C=26** pour afficher le personnage en haut à gauche du labyrinthe.

```
prgmLABY
5→L
26→C
prgmPERS
```

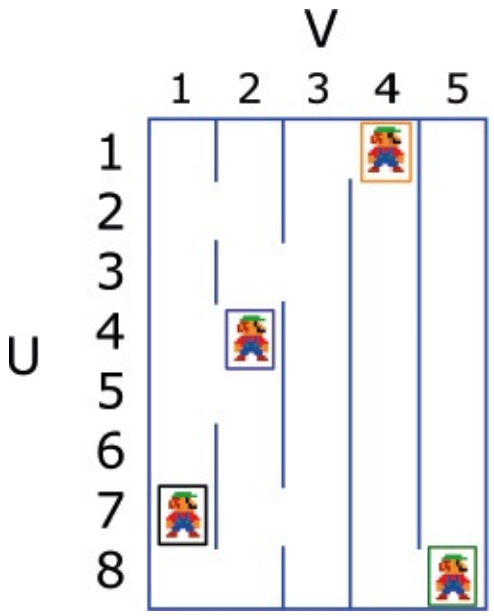
Pour gérer le déplacement du personnage, il faut savoir où ce dernier se positionne dans le labyrinthe. Considérons les cases (ligne **U**, colonne **V**) où il peut se situer.



go.eyrolles.com/ti-ch1ex18

Exercice 19

Déterminer la valeur de `U` et de `V` dans les quatre cas ci-après :



go.eyrolles.com/ti-ch1ex19

Initialisons les valeurs de **U** et **V** à 1 :

```
prgmLABY
5→L
26→C
prgmPERS
1→U
1→V
```

Puis prenons une variable **B** initialisée à 0 qui passera à 1 si le personnage atteint la case de la cerise. On va écrire une boucle **While B=0** et prendre la valeur de la touche enfoncée par l'utilisateur en la stockant dans une variable **A**.

Si l'utilisateur appuie sur la flèche (code 34), il faut vérifier si le personnage peut descendre. Cela n'est possible que si **U** est strictement inférieur à 8.

Si le personnage peut descendre, il faut alors effacer l'ancien personnage. Pour cela, il faut définir la valeur de **L** et **C** (qui servent à effacer le personnage) en fonction de **U** et **V**. On a prévu des cases de 22 pixels de large sur 20 pixels de haut, si bien qu'il est facile d'exprimer **L** et **C** en fonction de **U** et **V**.

En tenant compte de la valeur initiale du personnage (**L=5** et **C=26**), on obtient naturellement les relations :

L=5+20*(U-1) et **C=26+22*(V-1)**

On code alors cette affectation, puis on lance le programme **PERSOFF**.

On augmente **U** de 1, on définit la nouvelle valeur de **L** et on appelle le programme **PERS** pour afficher le personnage à son nouvel emplacement :

```
prgmLABY
5→L
26→C
prgmPERS
1→U
1→V
```

```
0→B
While B=0
getKey→A
If A=34
Then
If U<8
Then
5+20*(U-1)→L
26+22*(V-1)→C
prgmPERSOFF
U+1→U
5+20*(U-1)→L
prgmPERS
End
End
End
```

Exercice 20

Coder le cas où l'utilisateur appuie sur la touche (code 25).

Pour finaliser le mouvement du personnage, il faut vérifier s'il y a un mur à sa droite quand l'utilisateur appuie sur la touche (code 26). Si **V=**1, on teste la valeur de L_1**(U)** ; si **V=**2, on teste la valeur de L_2**(U)**, etc. Si la valeur est nulle, on autorise le déplacement. Voici donc la suite du programme :

```
If A=26
Then
If V=1
Then
If L1, (U)=0
Then
5+20*(U-1)→L
26+22*(V-1)→C
prgmPERSOFF
V+1→V
26+22*(V-1)→C
prgmPERS
End
End

If V=2
Then
If L2,(U)=0
Then
```

```
  5+20*(U-1)→L
  26+22*(V-1)→C
  prgmPERSOFF
  V+1→V
  26+22*(V-1)→C
  prgmPERS
  End
  End
```

go.eyrolles.com/ti-ch1ex20

Exercice 21

Traiter les deux autres cas où le personnage est en colonne 3 ou 4.

go.eyrolles.com/ti-ch1ex21

Exercice 22

Coder le cas où l'utilisateur appuie sur la flèche (code 24).

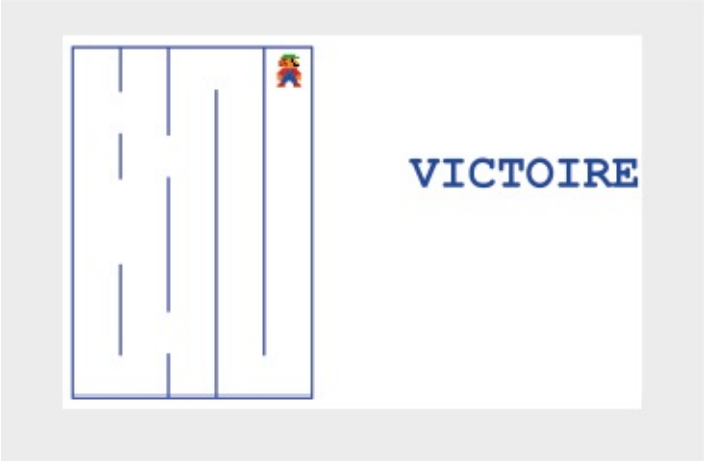
Pour finir, lorsque le personnage arrive dans la case du fruit, il faut programmer l'affichage de « VICTOIRE ». Pour cela, on pourra utiliser l'instruction **`Texte`** qui utilise les coordonnées des pixels (ligne, colonne) :

```
Texte(ligne, colonne,"Texte à écrire")
```

go.eyrolles.com/ti-ch1ex22

Exercice 23

Terminer le programme en prévoyant un test à la fin pour savoir si le personnage est au niveau de la cerise et empêcher tout autre déplacement.

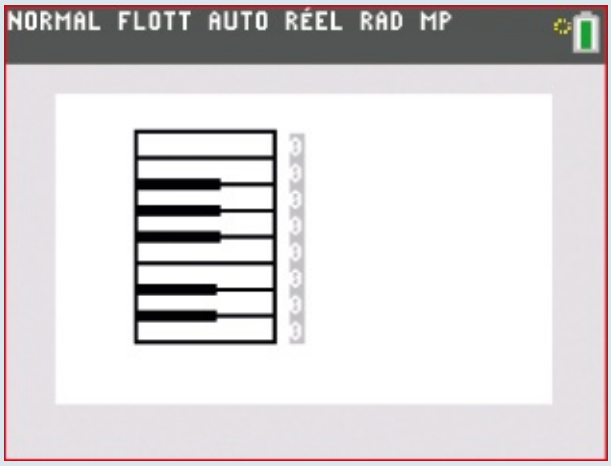


go.eyrolles.com/ti-ch1ex23

Chapitre 2

Le piano numérique

Le but de ce chapitre est de se familiariser avec le TI-Innovator™ Hub et d'utiliser ses capteurs intégrés, puis de le connecter à un capteur extérieur pour réaliser un piano téléguidé.



ÉTAPE 1 **Apprendre à coder une couleur et activer la LED du TI-Innovator™ Hub.**

ÉTAPE 2 **Déterminer la fréquence d'une note et jouer au piano avec le TI-Innovator™ Hub.**

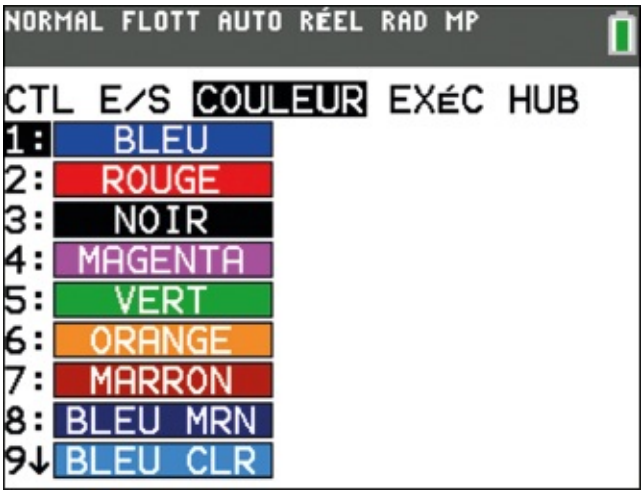
ÉTAPE 3 **Utiliser le capteur de lumière du TI-Innovator™ Hub.**

ÉTAPE 4 **Brancher un capteur de distance sur le TI-Innovator™ Hub.**

ÉTAPE 1 Et si on parlait couleurs ?

Sur la TI-83 Premium CE

Sur la TI-83 Premium CE, on dispose de 15 couleurs prédéfinies avec lesquelles on peut dessiner ou écrire du texte.



Chaque couleur peut être désignée par son nom, mais aussi par un numéro qui lui est associé. Ainsi, le bleu porte le numéro 10, le rouge le numéro 11, et ainsi de suite jusqu'à 24 (se référer à l'ordre d'apparition des couleurs pour en déduire ces numéros). Cette numérotation peut simplifier le codage, par exemple pour générer une image dont on fera varier la couleur dans une boucle.

Dans l'éditeur de programmes, le nom des couleurs est accessible en appuyant sur [prgm] et en choisissant l'onglet `COULEUR`.

Pour afficher un texte dans la fenêtre graphique, il faut utiliser l'instruction **`Texte(ligne, colonne,"texte à afficher")`**. Pour définir sa couleur, il faut précéder cette instruction de la ligne **`CouleurTexte(nom`** ou **`numéro de la couleur)`**.

Quand on écrit un programme, on accède à l'instruction `CouleurTexte` via [2nde], [dessin C prgm].

Ces deux programmes réalisent donc la même action :

```
CouleurTexte(ROUGE)
Texte(10,10,"COUCOU")
```

```
CouleurTexte(14)
Texte(10,10,"COUCOU")
```

Dans l'instruction `Texte`, le repérage utilisé est le pixel. Les coordonnées correspondent donc à la ligne et à la colonne du pixel.

À l'aide de l'instruction **`Ligne`**, accessible via [2nde], [dessin C / prgm], onglet **`DESSIN`**, dessinons maintenant un segment. Les arguments sont :

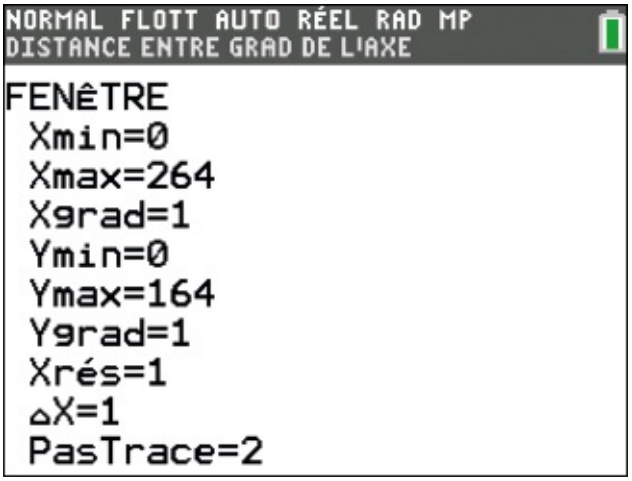
`Ligne(X1,Y1,X2,Y2,affichage, couleur)`

Une ligne s'affiche ainsi du point de coordonnées **`(X1,Y1)`** jusqu'au point **`(X2,Y2)`**. Ces coordonnées sont données dans le repère « classique » (voir plus bas). L'argument **`affichage`** peut prendre la valeur 0 pour effacer la ligne ou 1 pour l'afficher. Comme vu précédemment, la **`couleur`** doit être un numéro compris entre 10 et 24, ou bien le nom de la couleur.

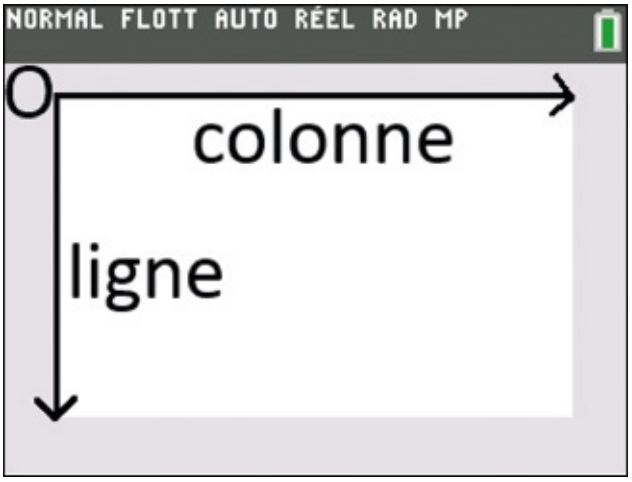
Comme vu dans le chapitre 1, il y a deux façons de repérer des points à l'écran :

- avec les coordonnées des pixels (origine en haut à gauche) ;
- avec le repère défini en appuyant sur la touche [fenêtre].

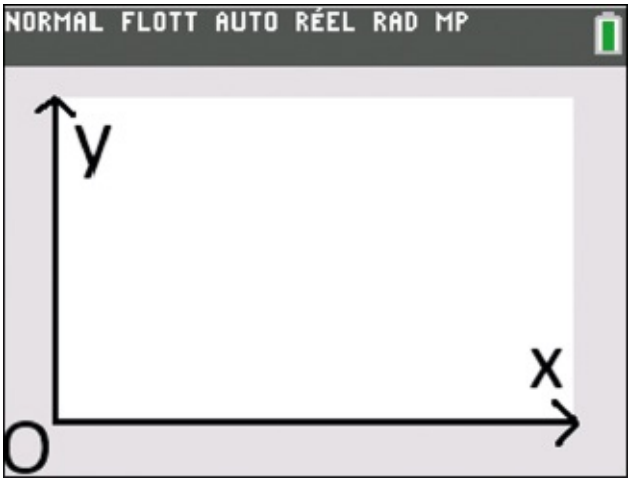
Afin de travailler facilement avec ces repères, il est conseillé de paramétrer la fenêtre ainsi :



L’origine du repère « pixel » est en haut à gauche :

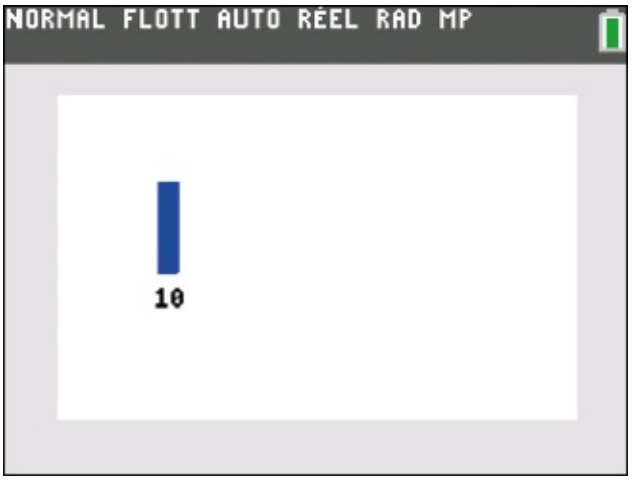


En revanche, l’origine du repère « classique » est en bas à gauche :

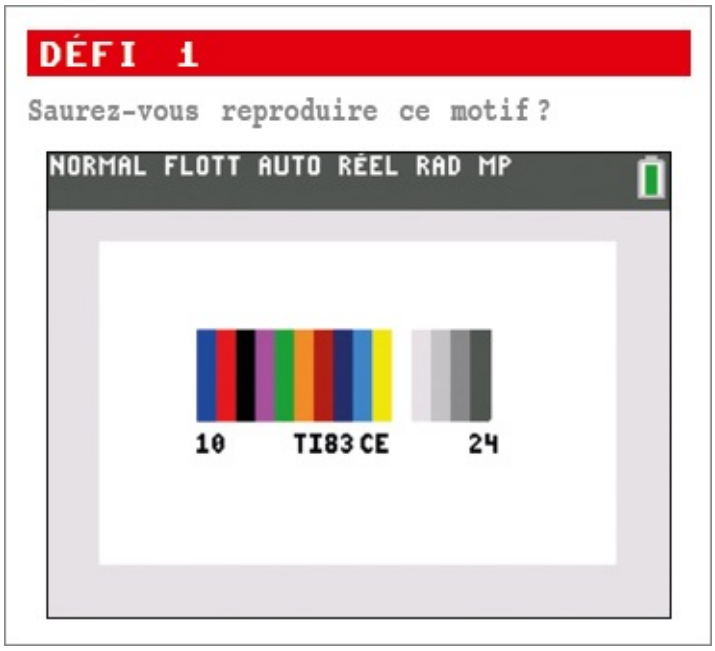


Exercice 1

Reproduire le rectangle ci-après, de largeur 10 pixels et de hauteur 45 pixels.



go.eyrolles.com/ti-ch2ex1



DÉFI 2
Et si on mettait un peu d'aléatoire ?
Comment créer un programme qui génère
cette ligne brisée ?
NORMAL FLOTT AUTO RÉEL RAD MP

Sur le TI-Innovator™ Hub

La TI-83 Premium CE permet de travailler avec 15 couleurs seulement. Mais il existe un système de couleurs bien plus complet : le code RGB (pour *Red Green Blue*, Rouge Vert Bleu ou RVB en français). En mélangeant ces trois couleurs « élémentaires » ou « primaires », on peut générer des millions d'autres couleurs. La valeur de chacune de ces trois composantes est un nombre compris entre 0 et 255.

En voici quelques exemples

COULEUR OBTENUE	R	G	B
	255	0	0
	0	255	0
	0	0	255
	255	0	255
	0	255	255

	128	128	128
	255	255	255
	0	0	0
	100	50	255

La vision humaine des couleurs est rendue possible grâce à des cônes de la rétine qui ont des sensibilités différentes au rouge, vert et au bleu.

Nous allons utiliser ce code RGB pour modifier la couleur de la LED multicolore intégrée au TI-Innovator™ Hub. Dans le code ci-dessous, l'utilisateur est invité à saisir les valeurs des composantes R, G et B, qui sont ensuite affectées à la diode du Hub.

Le TI-Innovator™ Hub est aussi appelé Hub.

```
EffÉcran
Disp "T=0 POUR ARRETER"
1→T
Repeat T=0
Prompt T,R,G,B
Send("SET COLOR eval(R) eval(G) eval(B) TIME 3")
Wait 3
End
```

`Prompt` permet à l'utilisateur de saisir les valeurs de `T`, `R`, `G` et `B` lors de l'exécution du programme. `TIME 3` signifie que la LED va rester allumée 3 secondes.

`Send( "SET COLOR` est obtenu en appuyant sur prgm, onglet `HUB`, et en choisissant `COLOR`.

La fonction **`eval`** est une des fonctions primordiales à connaître lorsqu'on utilise le TI-Innovator™ Hub. Ici, le programme envoie au Hub une ligne de commande, comprise entre les guillemets, qui contient une chaîne de caractères. La fonction

`eval` convertit la valeur contenue dans la variable passée en argument.

Supposons par exemple que 0, 128 et 255 soient affectés respectivement à R, G et B. La ligne **`Send("SET COLOR eval(R) eval(G) eval(B) TIME 3")`** enverra donc au Hub la commande suivante **`"SET COLOR 0 128 255 TIME 3"`**.

Si on n'utilise pas la fonction **`eval`** et que l'on écrit **`Send("SET COLOR R G B TIME 3")`**, le Hub recevra la commande **`"SET COLOR R G B TIME 3"`**, ce qu'il ne saura pas interpréter.

Exercice 2

Écrire un programme qui affiche une lumière orange, puis violette, puis blanche, pendant 1 seconde successivement.

`go.eyrolles.com/ti-ch2ex2`

ÉTAPE 2 Transformer sa calculatrice en piano

L'instruction SOUND

Le TI-Innovator™ Hub permet d'émettre des sons grâce à son haut-parleur intégré.

Le haut-parleur du Hub convertit le courant en son audible. Il peut émettre des sons sur des fréquences allant de 40 Hz à 4 000 Hz.

Pour utiliser ce haut-parleur, on doit employer l'instruction :

```
Send("SET SOUND Freq TIME temps")
```

`Freq` est un nombre correspondant à la fréquence en hertz et **`temps`** est exprimé en secondes.

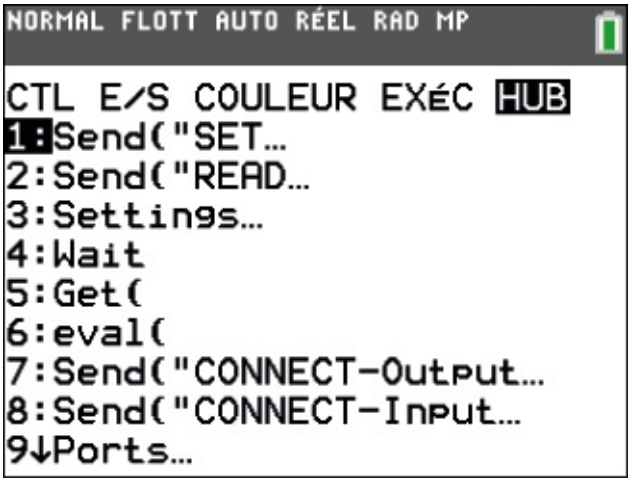
Lorsqu'un guitariste ou un pianiste accorde son instrument, il prend comme référence la note *la* qui doit vibrer à 440 hertz. Pour produire ce son pendant 1 seconde avec le Hub, on entre donc l'instruction :

```
Send("SET SOUND 440 TIME 1")
```

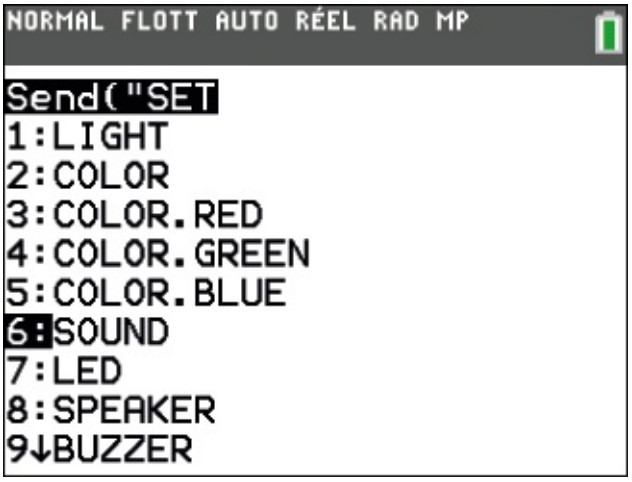
xs

Nous allons maintenant créer un programme pour émettre ce son.

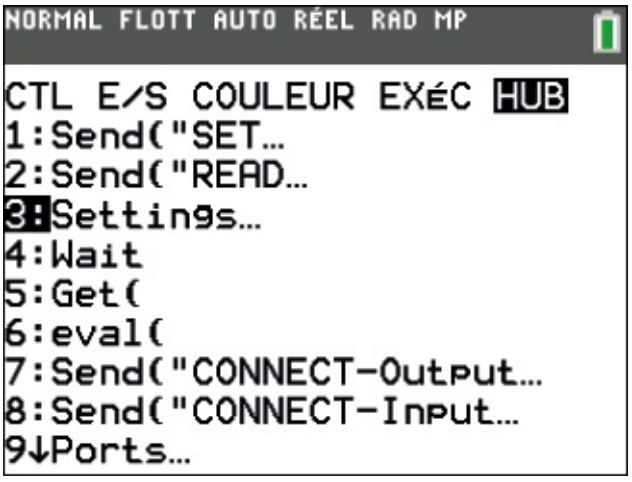
On commence par aller chercher l'instruction **`Send`**("**`SET`**, qui est accessible dans prgm, onglet **`HUB`**.

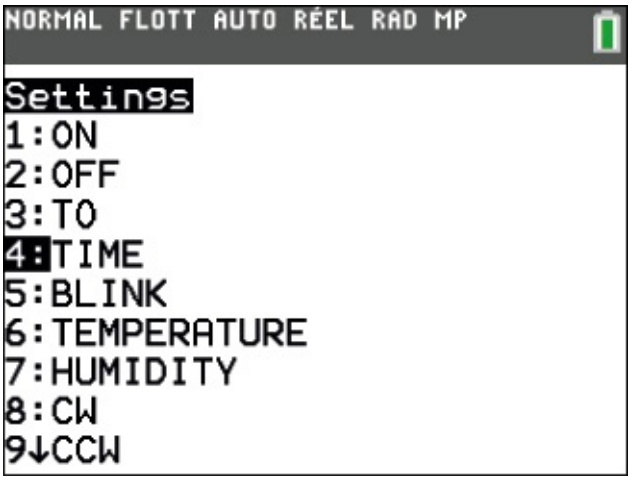


Une fois cette instruction sélectionnée, on choisit **SOUND**.

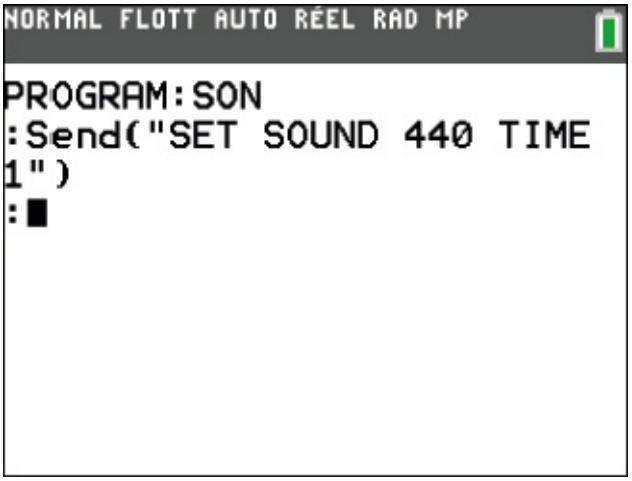


On entre alors la fréquence 440. On affiche ensuite l'instruction **TIME** en appuyant sur prgm, onglet **HUB**, et en choisissant **Settings** puis **TIME**.





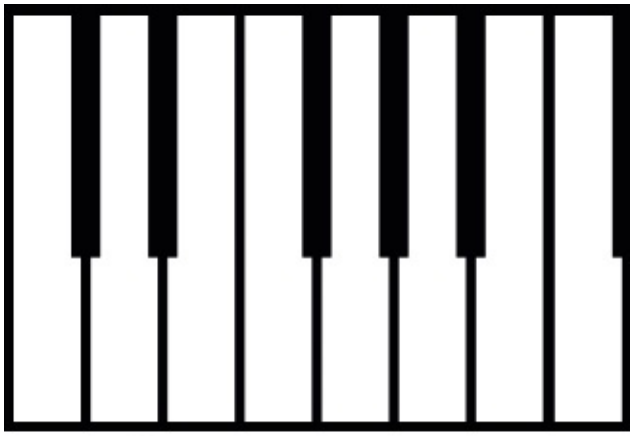
On a donc écrit la ligne :



En exécutant ce programme, on produit un son de 440 Hertz pendant 1 seconde.

La leçon de piano

Nous allons maintenant créer un petit clavier pour jouer du piano à l'aide des touches de la calculatrice ! Nous afficherons à l'écran l'image de ce clavier :



Il devra correspondre aux touches suivantes de la calculatrice :

Le travail à effectuer comporte trois étapes.

1. Dessiner le clavier avec un logiciel de dessin comme Paint ou Gimp.
2. Déterminer la fréquence de chacune des notes.
3. Associer à la frappe d'une touche l'émission du son correspondant à la fréquence de la note.

Les étapes 1 et 3 sont réalisables avec ce qui a été vu précédemment. Pour l'étape 2, nous avons besoin de quelques notions de musique.

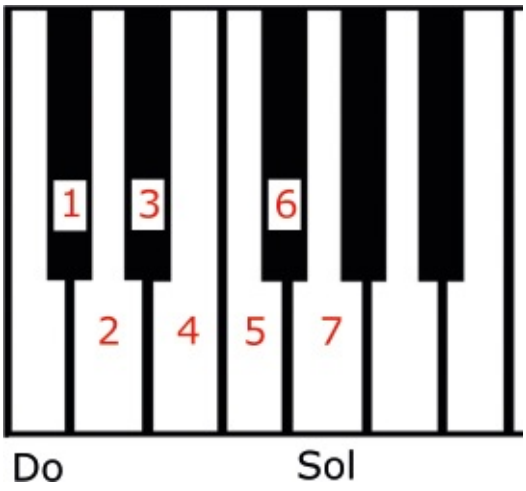
Pour connaître la fréquence de chaque note de notre gamme, nous allons faire appel aux fractions, en utilisant deux règles.

- Règle n° 1 : d'une octave à la suivante, les fréquences sont multipliées par 2.

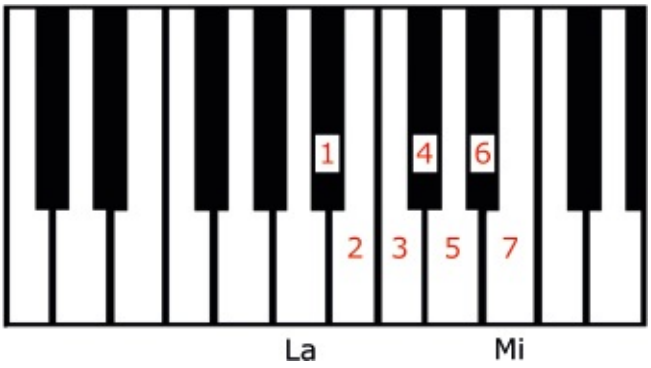
Par conséquent, comme le *la* de référence vibre à 440 Hz, le *la* de l'octave supérieure vibrera à 880 Hz et celui de l'octave inférieure à 220 Hz.

- Règle n° 2 : le rapport de fréquence d'une quinte est toujours de $\frac{3}{2}$ (pour ceux qui ne le savent pas, on précise qu'une quinte est composée de 7 demi-tons).

Dans la figure ci-dessous, on est parti du *do* et on a ainsi compté 7 demi-tons pour arriver au *sol*.



Autre exemple, on est parti ici du *la* et on a compté 7 demi-tons pour arriver au *mi*.

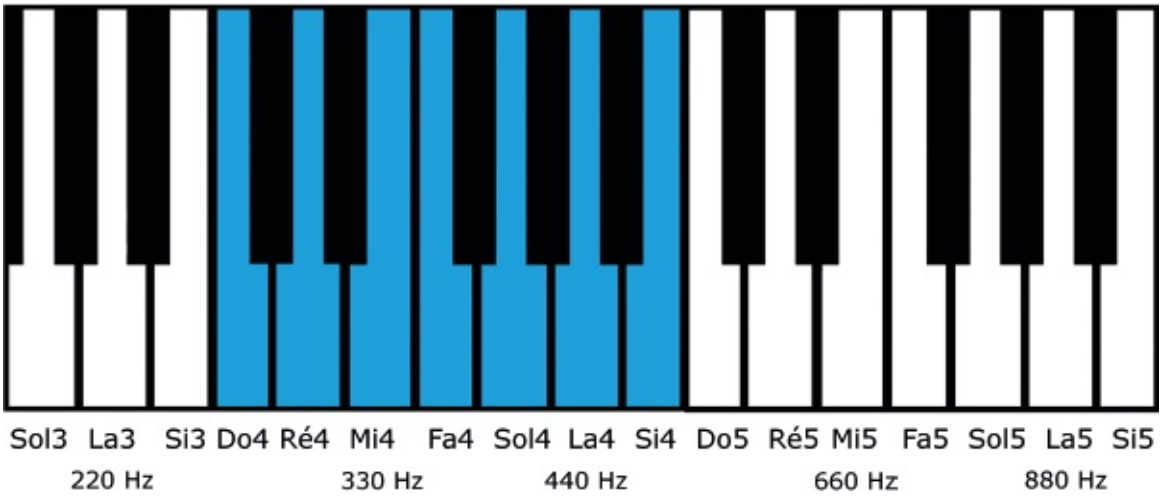


Si le *la* vibre à 440 Hz, le *mi* de la figure précédente, qui se trouve une quinte au-dessus, vibrera donc à :

$$\frac{3}{2} \times 440 = 660 \text{ Hz.}$$

Et le *mi* de l'octave précédente vibrera à $\frac{660}{2} = 330 \text{ Hz.} = 330$ Hz.

Afin de se repérer sur le clavier, numérotons les octaves. On a donc les résultats suivants :



Exercice 3

En poursuivant le travail de quinte en quinte et d'octave en octave, déterminer les fréquences de la quatrième octave (en bleu ci-dessus).

go.eyrolles.com/ti-ch2ex3

Exercice 4

Écrire à présent un programme `PIANO` qui associe à chaque touche une note de la gamme.

Indications : après avoir dessiné le clavier, on créera une boucle `While` et on prendra la valeur de la touche sur laquelle l'utilisateur appuie à l'aide de l'instruction `GetKey` qu'on stockera dans `A`. Puis, selon la valeur de `A`, on fera vibrer le haut-parleur du Hub en envoyant l'instruction `Send( "SET SOUND *** TIME 1" )`. On remplacera `***` par la fréquence qui convient (qui dépend donc de la valeur de `A`).

go.eyrolles.com/ti-ch2ex4

ÉTAPE 3 Le capteur de lumière

Sur le côté où est branché le câble reliant le TI-Innovator™ Hub à la calculatrice, on trouve un capteur de lumière qui permet de mesurer la luminosité *(Brightness)*.

L'instruction pour lire la quantité de lumière reçue par le capteur est **`Send("READ BRIGHTNESS")`**. Puis il faut récupérer cette valeur à l'aide de l'instruction **`Get(nomvariable)`**, qui la stockera dans une variable afin de l'exploiter.

> La commande `Send("READ BRIGHTNESS")` est accessible via prgm, onglet `HUB` : choisir alors `Send("READ`.

Stockons cette valeur de luminosité (exprimée en pourcentage) dans une variable **`B`** et commençons par l'afficher :

```
Send("READ BRIGHTNESS")
Get(B)
Disp B
```

> La commande `Get` est accessible via prgm, onglet `HUB`.

Ce code va donc afficher la valeur de la luminosité captée par le Hub au lancement du programme.

Modifions-le pour obtenir un affichage continu de la luminosité :

```
EffÉcran
0→A
While A=0
Send("READ BRIGHTNESS")
Get(B)
Output(5,5,B)
End
```

On crée une boucle infinie en affectant `0` à `A` et en écrivant `While A=0`.

À l'exécution du programme, on verra la valeur de la luminosité s'actualiser en continu. Si on approche le capteur d'une source de lumière comme une lampe, cette valeur augmentera.

On remarque que la valeur est plutôt inférieure à 10 dans une pièce. Mais elle augmente rapidement lorsqu'on s'approche d'une source lumineuse naturelle ou artificielle.

Exercice 5

Compléter les pointillés du programme ci-dessous, qui émet un son lorsque la valeur du capteur de lumière est supérieure à 10. À quoi sert la variable `L` ?

```
EffÉcran
0→A
0→L
While A=0
Send("READ BRIGHTNESS")
Get(B)
Output(5,5,B)
If B>……
Then
If L=0
Then
1→L
Send("SET SOUND 500 TIME 1")
End
End
End
```

go.eyrolles.com/ti-ch2ex5

DÉFI 3

Mettez le TI-Innovator™ Hub sous une lampe. Comment modifier le programme précédent pour que le Hub émette un son aigu quand la lampe est allumée et un son grave quand elle est

éteinte ?

ÉTAPE 4 Le capteur de distance

Nous allons à présent utiliser un capteur de distance, qui permet de mesurer la distance le séparant d'un obstacle à l'aide d'ultrasons. Il n'y en a pas de fourni avec le Hub, mais on peut s'en procurer un avec le kit TI-Innovator™ Breadboard Pack.

Tout d'abord, il faut connecter le capteur de distance au port IN 1 du TI-Innovator™ Hub.

Écrivons un programme qui affiche la valeur renvoyée par le capteur :

```
EffÉcran
1→A
Send("CONNECT RANGER 1 TO IN 1")
While A=1
Send("READ RANGER 1")
Get(R)
Output(5,13,100*R)
Wait 0.1
End
```

La ligne **`Send("CONNECT RANGER 1 TO IN 1")`** permet d'établir la connexion du capteur avec le TI-Innovator™ Hub au port IN 1. La ligne **`Send("READ RANGER 1")`** écrit la valeur renvoyée par le capteur et **`Get(R)`** stocke cette valeur dans la variable **`R`**. On a écrit **`100*R`** pour exprimer la valeur en centimètres.

Exécutez ce programme et mettez votre main au-dessus du capteur en l'approchant ou en la reculant.

Exercice 6

Modifier le programme précédent pour qu'un son soit émis si votre main est à moins de 20 cm du capteur.

go.eyrolles.com/ti-ch2ex6

Exercice 7

Jean-Michel Jarre utilise des capteurs pour jouer de la musique dans ses concerts : il met sa main au-dessus d'un capteur ce qui génère une note. Reproduisez ce phénomène en écrivant un programme qui joue un do si la distance main-capteur est inférieure à 10 cm, un ré si elle est comprise entre 10 et 15 cm, et un mi entre 15 et 20 cm.

go.eyrolles.com/ti-ch2ex7

DÉFI 4

Serez-vous capable de créer un programme affichant un carré de 3 × 3 pixels qui se déplace sur les ordonnées en fonction de la distance main-capteur ? Le but sera de colorier entièrement la ligne. Attention, l'instruction `Pxl-Aff` ne fonctionnant qu'avec des valeurs entières, il faudra utiliser la fonction `ent` qui permet d'isoler la partie entière d'une variable.

Pour voir une vidéo du résultat : go.eyrolles.com/ti-ch2de4

Si vous possédez un deuxième capteur de distance, modifiez le programme précédent afin de vous déplacer selon les abscisses et utilisez vos deux mains pour colorier tout l'écran.

Chapitre 3

Le jeu de basket

Le but de ce chapitre est de programmer un jeu de lancer de ballon de basket qui comptabilise le nombre d'essais nécessaires pour marquer un panier. Ce jeu se jouera seul, mais pourra être étendu à deux joueurs.

ÉTAPE 1 **Préparer les images d'arrière-plan de début et de fin de jeu afin d'accélérer l'affichage graphique.**

ÉTAPE 2 **Décrire les fonctionnalités du jeu et tracer les dessins du panier et des jauges.**

ÉTAPE 3 **Coder la boucle principale du jeu et tracer la trajectoire du ballon.**

ÉTAPE 1 Préparation des images d'arrière-plan

On a pu constater dans le chapitre 1 que les dessins mettent du temps à s'afficher point par point sur l'écran de la calculatrice. Heureusement, tous les éléments graphiques de notre jeu n'ont pas besoin d'être créés en temps réel : certains d'entre eux peuvent être dessinés à l'avance, comme on va le voir. Sinon, à chaque exécution de l'instruction **`EffDess`**, il faudrait tout redessiner, en particulier les éléments d'arrière-plan, ce qui se traduirait par un temps d'attente désagréable pour le joueur.

Nous allons d'abord dessiner l'écran de lancement du jeu, appelé *splash screen*. Son but est de présenter le thème du jeu au joueur, mais pas seulement… Cette image devra faire 165 pixels de hauteur et 265 pixels de largeur.

Exercice 1

Avec un outil de dessin, dessiner sur ordinateur un écran de démarrage pour notre jeu de basket. Puis transférer sur la calculatrice le fichier image de cet écran via le logiciel TI-Connect™ CE. Nommer ce fichier `Image8`.

`go.eyrolles.com/ti-ch3ex1`

Passons maintenant à l'écriture du programme ou plutôt des deux programmes associés. Car un splash screeen a un deuxième rôle, celui d'initialiser l'environnement graphique du jeu. Le temps que cette opération se réalise, il

occupe l'attention du joueur au lieu de le laisser devant un écran noir.

Exercice 2

Dans le programme `DINIT` ci-dessous, identifier le rôle de chaque ligne de code.

> **Le nom du programme commence par un D pour indiquer qu'il s'agit d'un programme de dessin.**

```
Radian
Fonc
FoncNAff
GraphNAff
QuadNAff
ÉtiqNAff
PleinÉcr
CGRect
1→Xmin
265→Xmax
0→Xgrad
1→Ymin
165→Ymax
0→Ygrad
DispGraph
EffDess
ArrPlanNAff
```

DÉFI 1

Si on supprime l'affichage des axes du repère sur la calculatrice, il faudra bien les faire réapparaître lorsqu'elle sera utilisée en cours de mathématiques ! Une remarque qui vaut aussi pour les autres variables d'environnement graphique. Heureusement, avant de modifier un environnement, il est possible de le sauvegarder via l'instruction `EnrBDG`. Revoyez alors le programme `DINIT` en rappelant l'environnement à la fin (grâce à l'instruction `RappelBDG`).

Le programme **`DINIT`** a donc effacé tout le contenu graphique de la calculatrice et configuré l'environnement pour que l'on puisse démarrer avec une page blanche. Nous allons à présent remplir cette page avec l'écran de lancement déjà dessiné.

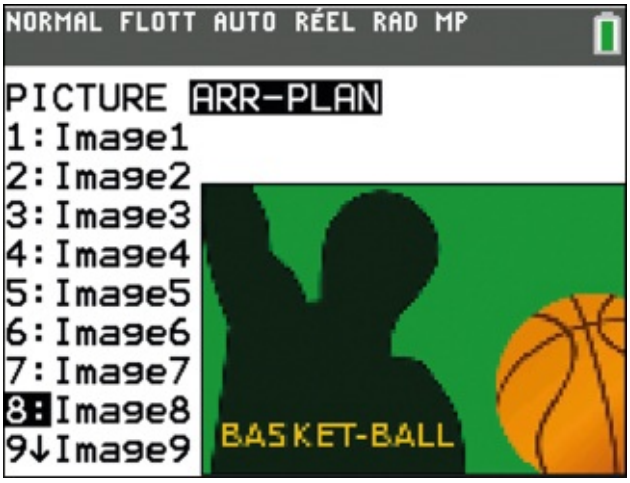
go.eyrolles.com/ti-ch3ex2

Écrivons le programme **DSPLASH** :

```
prgmDINIT
ArrPlanAff 8
Output(4,4,"LANCEMENT")
CouleurTexte(JAUNE)
Texte(25,125,"APPUYER SUR ENTRER")
```

Le nom d'un programme ne doit pas comporter plus de 8 caractères.

L'image de l'écran de démarrage a été importée en 8e position dans la calculatrice, qui peut stocker en tout 10 images d'arrière-plan. Il est donc important de mémoriser la place de l'image que l'on souhaite utiliser dans le programme. Pour vérifier la position de cette image, appuyez sur la touche var. Allez alors dans le menu **Pic et arrière-plan** et naviguez parmi les différentes images.



Exercice 3

Déterminer la différence entre les instructions `Output` et `Texte`.

`go.eyrolles.com/ti-ch3ex3`

Exercice 4

Coder le programme `DFINAL` pour qu'il affiche un écran final de victoire si le joueur a inscrit un panier, en indiquant le nombre d'essais (stocké dans la variable `N`) qui auront été nécessaires pour marquer. L'image de cet écran devra être importée en 7e position dans la calculatrice.

Voici un exemple d'écran final, où le score correspond au nombre de coups joués.

Si le joueur a marqué, le programme `DFINAL` demandera au joueur de presser la touche [entrer], effacera les dessins présents à l'écran (trajectoire du ballon, panier, etc.) et affichera ce nouvel arrière-plan de victoire. Le jeu s'arrêtera une fois la touche [entrer] pressée de nouveau.

Ce travail préliminaire nous a permis d'appréhender quelques fonctions graphiques essentielles à la réalisation d'un jeu, même si habituellement cette phase de dessin intervient plutôt à la fin.

`go.eyrolles.com/ti-ch3ex4`

ÉTAPE 2 Description du jeu et tracé des dessins

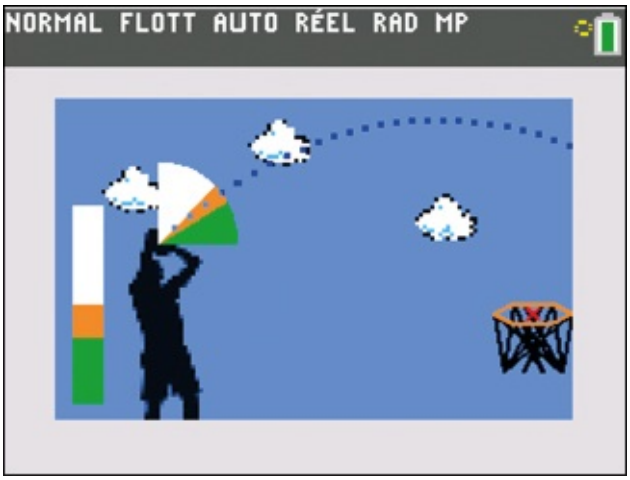
Commençons par décrire le fonctionnement du jeu.

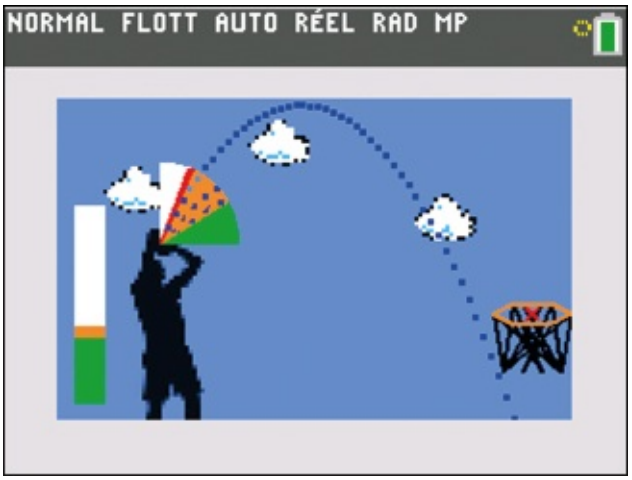
Le ballon sera toujours lancé à partir du même endroit de l'écran, au niveau de la

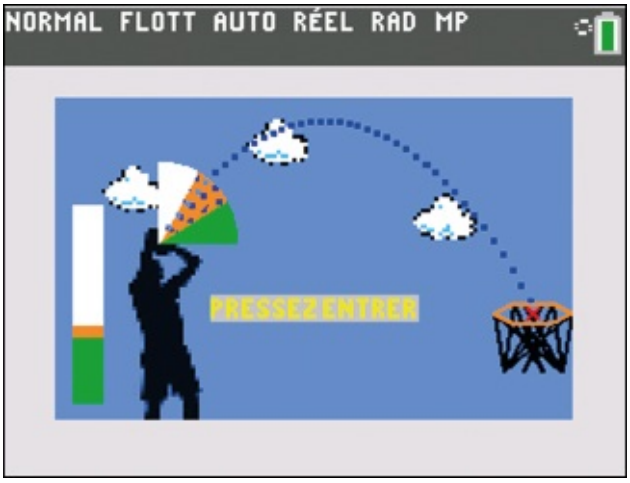
main du joueur dessiné en train de sauter. Ce sera en fait le panier qui se déplacera d'une partie de jeu à l'autre. Il faudra donc stocker les coordonnées de ce panier pour que le programme puisse déterminer si le ballon l'a ou non traversé.

L'angle et la force du tir seront représentés respectivement par une jauge en quart de cercle et une jauge verticale. Si on souhaite améliorer le jeu, on pourra faire évoluer la couleur des jauges du vert au rouge en fonction de la valeur de l'angle et de la force.

Pour vous aider à comprendre ce que vous allez coder, voici quelques captures d'écran du jeu :







S'il y a bien quelque chose dont on ne manque pas sur la calculatrice, ce sont des touches ! Nous allons en utiliser 8 dans notre jeu.

- La touche [entrer] permettra de valider un certain nombre d'écrans.
- Les flèches ◀ et ▶ régleront l'angle de tir du lancer du ballon.
- Les flèches ▲ et ▼ régleront la force du tir.
- La touche [alpha] permettra d'activer ou de désactiver un mode turbo pour le remplissage des jauges.
- La touche [graphe] servira à lancer le ballon.
- La touche [mode] permettra de forcer la fin du jeu.

Comment le programme du jeu va détecter si l'une de ces touches est pressée par le joueur ? Pour la touche [entrer], il suffira d'introduire la fonction **`Pause`** dans le code, qui bloquera le programme jusqu'à ce que cette touche soit enfoncée. Pour les 7 autres, on utilisera leur code et la fonction **`GetKey`** (voir chapitre 1).

Exercice 5

Déterminer les codes de ces 7 touches.

go.eyrolles.com/ti-ch3ex5

Exercice 6

Dessiner à présent sur ordinateur le fond d'écran de votre jeu, sur lequel devra aussi apparaître le joueur. Ici, nous avons choisi un ciel bleu avec un basketteur en train de sauter, mais libre à vous de représenter un parquet et des gradins remplis de supporters !

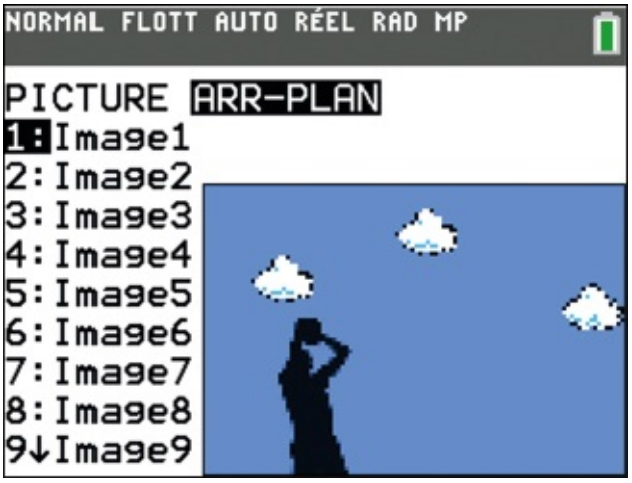
> **Pour l'image de fond, il est impératif de respecter la taille déjà mentionnée auparavant (165 × 265 pixels).**

Seuls le panier, les jauges et la trajectoire du ballon vont être dessinés par le programme principal.

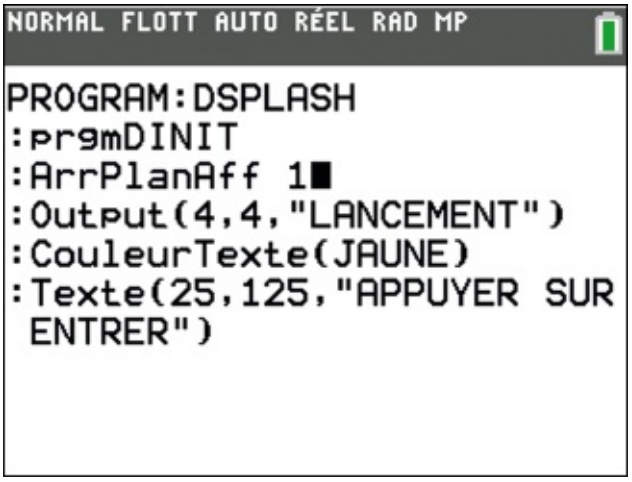
go.eyrolles.com/ti-ch3ex6

Position de la main

Pour commencer, il faut repérer la position de la main du joueur à partir de laquelle le ballon sera lancé. Pour cela, une fois l'image de fond chargée dans la calculatrice, modifiez temporairement le programme **`DSPLASH`** en remplaçant le n° de fond d'écran à afficher par celui de votre fond de jeu. Ici, notre fond de jeu se trouve en 1re position.

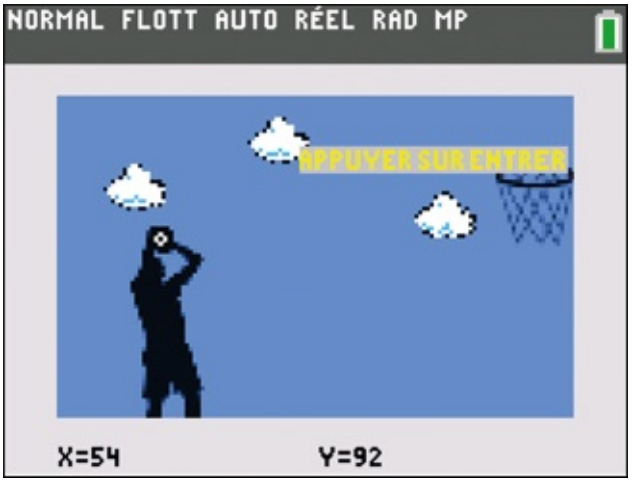


On modifie donc le programme en conséquence :



Puis exécutez-le : **DSPLASH** appelle alors le programme **DINIT**, puis charge l'image de fond de jeu et s'interrompt. À l'aide des flèches de la calculatrice, déplacez-vous jusqu'à la position de la main et relevez ses coordonnées. Enfin, corrigez le programme **DSPLASH** en y replaçant le fond d'écran initial.

Sur notre image, la main se positionne en **(54;92)**. Nous avons choisi de coder « en dur » ces coordonnées dans le programme plutôt que d'utiliser deux variables supplémentaires.



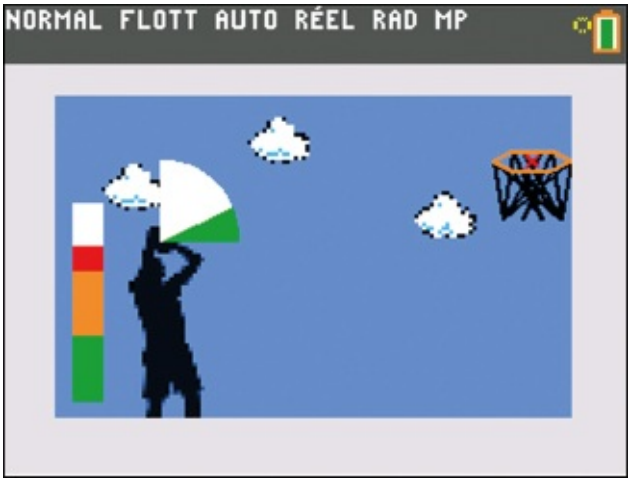
Pour nos tests, nous avons dessiné un panier sur le fond de jeu dans un premier temps. Mais ce panier est à supprimer puisqu'il sera dessiné par le programme lui-même. Nous travaillerons donc maintenant avec ce fond d'écran :

Le panier de basket et les jauges

Exercice 7

Coder le programme `DPANIER` qui dessine le panier de basket. La position du centre du panier (la croix rouge sur l'image) sera initialisée dans le programme principal. Son abscisse devra être stockée dans la variable `L` et

son ordonnée dans la variable `H`. Le dessin du panier sera réalisé à l'aide d'appels répétés à la fonction `Ligne` (voir chapitre 2**).**



Il nous reste maintenant à dessiner les jauges du jeu.

- La force du tir sera stockée dans la variable `F`. Elle sera comprise entre 0 et 100, et initialisée à 100.
- L'angle de tir sera stocké dans la variable `R`. Il sera compris entre 0 et 90, et initialisé à 90.
- Le pas d'incrément des jauges sera stocké dans la variable `Z`. Il vaudra 1 ou 10 selon que le mode turbo est désactivé ou non, et il sera initialisé à 1.

`go.eyrolles.com/ti-ch3ex7`

Exercice 8

Coder le programme `DJAUGE` qui dessinera la jauge verticale symbolisant la force du tir.

Le placement de cette jauge dépendra de votre décor. Dans notre programme, nous l'avons représentée de la position `(10;10)` pour son coin inférieur gauche jusqu'à la position `(25;110)` pour son coin supérieur droit.

Sa hauteur dépend quant à elle de la variable `F`. Nous avons choisi de compléter toute la partie vide de la jauge par du blanc.

Ce programme `DJAUGE` ne fait que dessiner la jauge en fonction de la valeur de `F`. Il ne gère pas la saisie des touches ni donc la modification de `F`. C'est la boucle principale du jeu qui s'en chargera.

`go.eyrolles.com/ti-ch3ex8`

Exercice 9

Codez le programme `DANGLE` qui dessinera la jauge représentant l'angle du tir.

À vous de choisir la position de cette jauge en quart de cercle. Ici, le cercle a pour centre la position de la main `(54;92)` et pour rayon 40 pixels. La hauteur de la jauge dépend de la variable `R`. Nous avons choisi de compléter par du blanc toute la partie vide de la jauge.

Ce programme ne fait que dessiner la jauge en fonction de la valeur de `R`. Il ne gère pas la saisie des touches ni donc la modification de `R`. C'est la boucle principale du jeu qui s'en chargera.

`go.eyrolles.com/ti-ch3ex9`

Test intermédiaire

À ce stade du projet, vous avez probablement envie de voir ce que cela peut donner ! Créez le programme `TEST` en saisissant le code suivant (on expliquera plus loin ce que représentent les variables `G`, `M` et `K`) :

```
prgmDSPLASH
1→G
100→F
90→R
0→M
0→K
1→Z
0→N
nbrAléatEnt(50,150)→H
245→L
Pause
prgmDBACKGR
```

où **prgmDBACKGR** contient lui-même les éléments de dessin du jeu, à savoir :

```
prgmDINIT
ArrPlanAff 1
EffDess
EffÉcran
prgmDJAUGE
prgmDANGLE
prgmDPANIER
```

Désormais, notre jeu s'initialise et se dessine. Nous allons maintenant coder les actions du joueur en complétant l'écriture du programme **TEST**.

ÉTAPE 3 Codage de la boucle principale du jeu

Une fois cette phase d'initialisation validée, il faut à présent tester en boucle si le joueur règle l'angle ou la force de son tir, réalise une tentative de tir, ou souhaite interrompre la partie. Puis nous devrons modifier la jauge ou l'angle de tir, ou effectuer le tir du ballon.

Pour déterminer quelle action graphique doit réaliser la TI-83 Premium CE, introduisons une variable **G**. Nous allons écrire une boucle qui s'arrête lorsque **G** vaut 4 et qui stocke dans une variable **K** le code de la touche pressée par le joueur.

Exercice 10

Quelle est la différence entre REPEAT et WHILE ?

Si **G** vaut 1, sa valeur initiale, le joueur paramètre son tir, ce qui consiste à incrémenter ou décrémenter les valeurs de **F** ou **R** en fonction des touches utilisées.

De plus, le programme doit savoir s'il doit actualiser l'affichage de la jauge de force ou celui de la jauge d'angle. Pour cela, on va introduire la variable **M**, qui vaudra 1 si c'est la jauge de force qui doit être modifiée, et 2 si c'est la jauge d'angle.

go.eyrolles.com/ti-ch3ex10

Exercice 11

Compléter les pointillés du code ci-dessous dans le cas où le joueur appuie sur la touche ▲. Attention, il ne faut pas oublier que la variable F ne doit pas dépasser 100, donc il faut prévoir une instruction pour en tenir compte.

```
If K=……
Then

If G=1
Then
……
……
……
……

……→M
End
End
```

go.eyrolles.com/ti-ch3ex11

Exercice 12

En s'inspirant du programme précédent, écrire à la suite la portion de code correspondant au cas où le joueur presse la touche . Attention, ne pas oublier que `F` ne peut pas descendre en dessous de 0 !

go.eyrolles.com/ti-ch3ex12

Exercice 13

De même, écrire à la suite la portion de code traduisant le cas où le joueur appuie sur la touche , puis le cas où il presse . Attention, `R` ne peut pas dépasser 90 ni être inférieur à 0 !

Maintenant, si **`M=1`**, on lance le programme qui actualise la jauge de force :

```
If M=1
Then
prgmDJAUGE
0→M
End
```

Et si **`M=2`**, on actualise l'affichage de la jauge d'angle :

```
If M=2
Then
prgmDANGLE
0→M
End
```

go.eyrolles.com/ti-ch3ex13

Exercice 14

Coder l'action de la touche alpha qui active ou désactive le mode turbo.

Lorsque le joueur a fini son paramétrage de tir, il appuie sur la touche graphe. La variable `G` doit alors prendre la valeur 3.

> **Pourquoi 3 et pas 2 ? Parce qu'on se réserve cette valeur 2 dans le cas d'une déclinaison du jeu pour deux joueurs. L'état `G=2` permettrait alors de gérer l'alternance entre les joueurs.**

go.eyrolles.com/ti-ch3ex14

Exercice 15

Compléter les pointillés du code ci-dessous dans le cas où le joueur presse la touche graphe :

```
If K=……
Then
…… →G
End
```

go.eyrolles.com/ti-ch3ex15

Si **`G=3`**, il faut alors dessiner la trajectoire du ballon. On va pour cela écrire le programme **`DTRAJECT`**. Le calcul de cette trajectoire et son affichage sont assez difficiles…

Nous allons utiliser ce qu'on appelle une « représentation paramétrique » de la trajectoire à l'aide de la fonction **Pt-Change(X,Y,Couleur)**. Cette fonction allume le pixel de coordonnées **(X;Y)** avec le paramètre **Couleur**. Deux possibilités : soit le pixel était « éteint » et dans ce cas la procédure est réalisée ; soit il était déjà allumé et dans ce cas la fonction éteint le pixel, laissant apparaître celui de l'image de fond.

Cette fonction est donc intéressante car elle n'interagit qu'avec l'écran graphique et ne touche pas à l'image de fond. Ainsi, le ciel et ses nuages resteront intacts, à la différence de la jauge d'angle si elle est touchée par le dessin de la trajectoire du tir.

De plus, en appelant deux fois de suite le programme **DTRAJECT**, on peut effacer la trajectoire du tir pour préparer le tir suivant.

Attention, l'équation de la trajectoire dépend de la position de départ du ballon. Si la main n'est pas située comme ici aux coordonnées (54;92), il faudra modifier cette équation dans le programme DTRAJECT.

Pour vous aider, vous pouvez télécharger le programme **DTRAJECT** :

go.eyrolles.com/ti-ch3DT

Vous pouvez aussi consulter le lien **https://education.ti.com/fr/activities/france/algorithme/programmgraphique-en-couleur**

Le jeu est bientôt terminé !

Si la variable **G** vaut 3, le joueur a fini son paramétrage de tir et a appuyé sur la touche 15 (touche graphe). Dans ce cas, le programme incrémente le nombre de

coups joués, stocké dans la variable **N**. Il appelle ensuite **DTRAJECT** pour dessiner la trajectoire du tir. Si le tir est réussi, **DTRAJECT** passe la valeur de **G** à 4. Si le tir est raté, **G** vaut toujours 3 et le programme appelle de nouveau **DTRAJECT** pour effacer la trajectoire du tir. De plus, si **G** est différent de 4, on repasse en mode « paramétrage de tir » où **G** vaut 1.

Exercice 16

Compléter les pointillés du code ci-dessous pour réaliser les actions précédentes.

```
If G=3
Then
……→N
prgmDTRAJECT
If G=3
Then
prgmDTRAJECT
End
If G≠ ……
Then
1→G
End
End
End
prgmDFINAL
```

Pour rappel, N représente le nombre de lancers.

go.eyrolles.com/ti-ch3ex16

Le jeu doit s'arrêter lorsque le joueur appuie sur la touche mode ou marque un panier. Il faut alors affecter à **G** la valeur 4.

Exercice 17

Compléter les pointillés du code ci-dessous correspondant au cas où la touche [mode] est pressée :

```
If K=……
Then
……→G
End
```

Le cas où le joueur marque un panier est détecté lors du tracé de la trajectoire, qui est codé dans le programme **DTRAJECT**.

go.eyrolles.com/ti-ch3ex17

La programmation du jeu est finie : tous les éléments qui composent la boucle de jeu principal ont été traités. Pour vous aider, ils ont été réunis dans un seul programme **DSAISIE** que vous pouvez télécharger.

go.eyrolles.com/ti-ch3DS

Ainsi, en partant de ce moteur de jeu, vous pourrez essayer de relever le défi suivant…

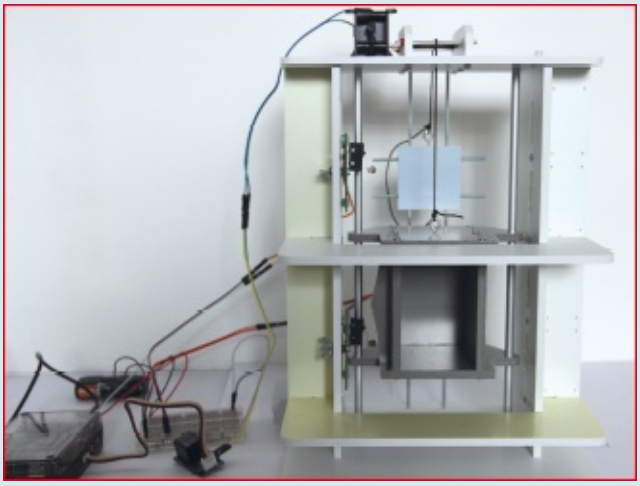
DÉFI 2

Et si vous amélioriez encore le programme de ce jeu ? En affichant un tableau des scores, en limitant le nombre de lancers, ou même en adaptant le jeu pour deux joueurs…

Chapitre 4

L’ascenseur

Le but de ce chapitre est de fabriquer un ascenseur en repoussant les limites d’interaction et de pilotage de la calculatrice avec le TI-Innovator™ Hub.



ÉTAPE 1 **Apprendre à utiliser une platine d’expérimentation et y connecter des microrupteurs.**

ÉTAPE 2 **Découvrir comment faire fonctionner un relais et un moteur sur la platine.**

ÉTAPE 3 **Fabriquer la cage d’ascenseur.**

ÉTAPE 4 **Finaliser les branchements du montage et écrire les programmes associés.**

ÉTAPE 5 **Gérer la descente de la cabine d’ascenseur.**

Matériel nécessaire au projet

Pour réaliser le projet de ce chapitre, vous aurez besoin de vous procurer les composants et équipements suivants :

- deux microrupteurs ;
- un relais de la marque Grove compatible avec le TI-Innovator™ Hub ;
- une alimentation externe 5 V USB ;
- une alimentation 3 V connectable sur la platine d'expérimentation ;
- un moteur à courant continu compatible 3 V ;
- une platine d'expérimentation ;
- des câbles pour platine d'expérimentation.

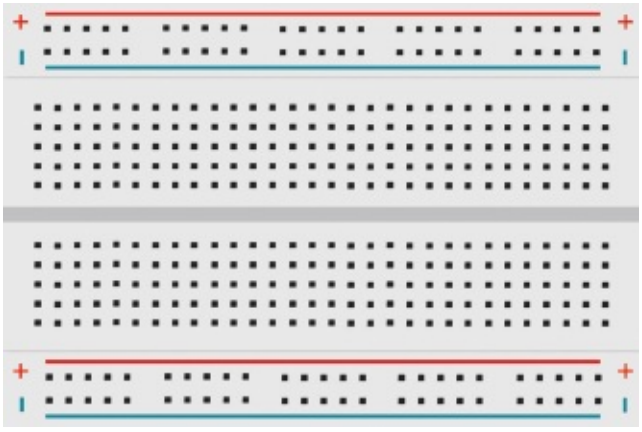
Une partie de ce matériel figure dans le kit TI-Innovator™ Breadboard Pack.

Vous aurez aussi besoin de fournitures pour fabriquer une cage d'ascenseur, sauf si vous l'achetez en kit.

ÉTAPE 1 Connexion des microrupteurs

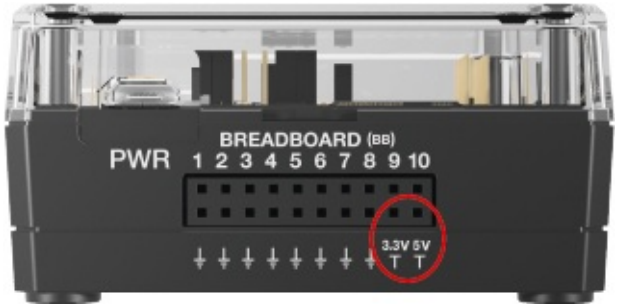
Tout développeur qui souhaite apprendre à piloter moteurs et capteurs sera amené tôt ou tard à utiliser une platine de prototypage, appelée encore *breadboard* (voir préambule du livre). Le principe en est simple : on vient directement enficher ses composants sur la plaque selon le schéma électronique qu'on a imaginé et que l'on souhaite valider.

Sur la platine ci-après, on rappelle que les points contenus sur chaque ligne + sont interconnectés, de même que ceux d'une ligne -. En revanche, ces lignes sont isolées les unes des autres, ce qui permet notamment d'obtenir deux niveaux d'alimentation (5 V et 12 V, par exemple) pour certains montages. Quant aux points en colonnes, ils sont reliés les uns par autres par colonnes de cinq. C'est sur ce type de platine que nous allons réaliser les différents montages de ce chapitre.



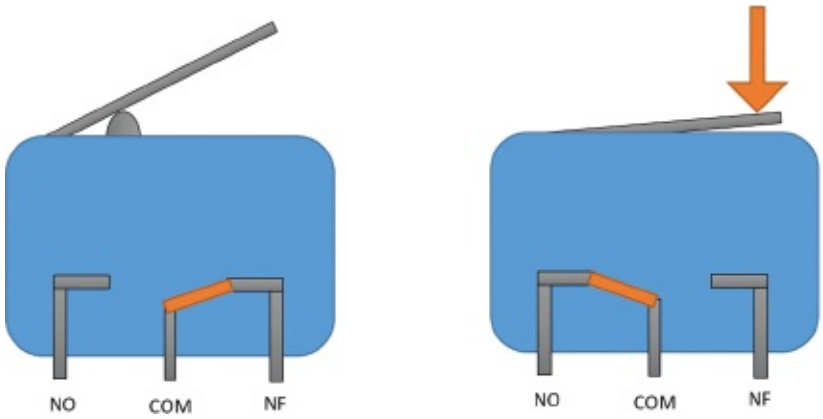
Dans un premier temps, pour délivrer une tension aux composants électroniques utilisés sur la platine d'expérimentation, nous utiliserons les ports marqués 3.3V et 5V du Hub. Par la suite et par sécurité, nous aurons recours à une alimentation secondaire pour le moteur de l'ascenseur de notre projet.

Si on prévoyait une seule alimentation pour l'ensemble du projet, on risquerait de solliciter trop d'ampérage et de griller le Hub. En isolant l'alimentation du moteur dans un circuit dédié, on réduit quasiment ce risque à néant.



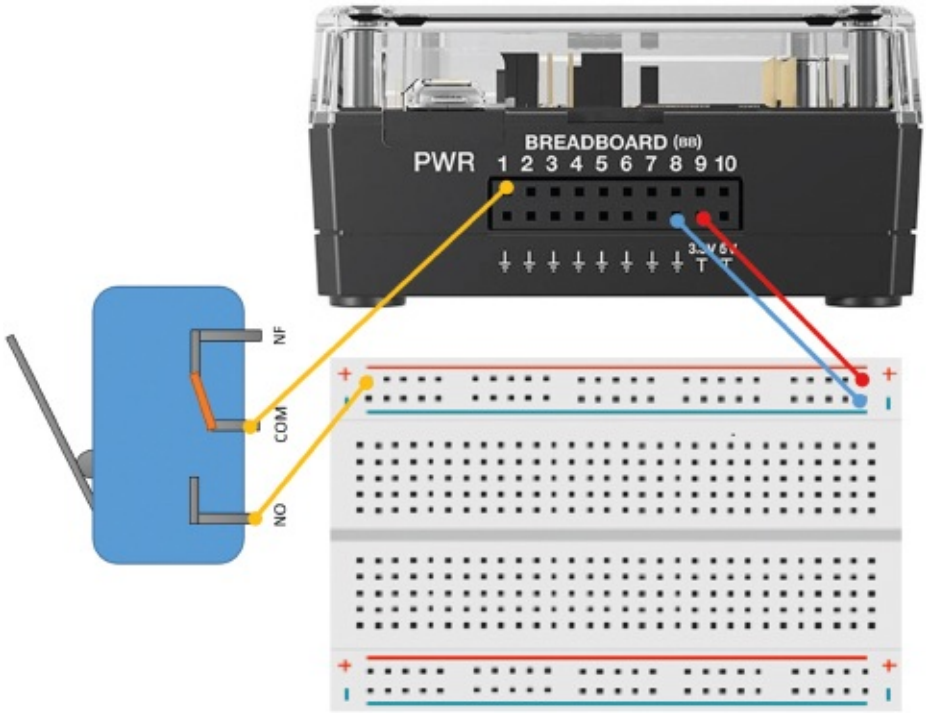
Pour nous ouvrir sur l'extérieur à l'aide des ports breadboard (BB) du Hub, nous allons relier ce dernier à un composant électronique appelé microrupteur. On lira en boucle l'état (ouvert ou fermé) de ce composant en l'affichant sur la calculatrice.

Un microrupteur possède trois broches. S'il n'est pas enclenché, les broches COM et NF sont reliées entre elles. Si un contact est établi, l'interrupteur est fermé et ce sont alors les broches NO et COM qui sont interconnectées. C'est ce dernier événement que nous allons détecter.



Attention ! Pour que le TI-Innovator™ Hub puisse détecter cet événement, il faut que l'enclenchement du microrupteur délivre une tension sur l'un des ports du Hub, par exemple le port BB1. Par conséquent, il ne suffit pas de relier directement les broches NO et COM aux broches BB1 et Masse puisque aucune tension ne serait fournie. On doit auparavant récupérer la tension de sortie du Hub de 3,3 V sur la platine, puis la réinjecter dans le Hub via le microrupteur.

On réalise donc le montage ci-après, où la fermeture de l'interrupteur permettra bien de délivrer 3,3 V sur la broche BB1 du Hub.



Écrivons maintenant le programme qui va nous permettre de détecter la fermeture du microrupteur. Le principe est toujours le même : on initialise d'abord le Hub en décrivant les composants électroniques connectés à ses différents ports (dans ce cas, le port BB1 uniquement), puis on développe la boucle principale. Ici, nous allons nous servir d'une entrée numérique qui renvoie soit 1 soit 0 selon qu'un événement se produit ou pas.

Saisissez le programme suivant :

```
Send("CONNECT DIGITAL.IN 1 TO BB1")
0→T
While T≠1
Send("READ DIGITAL.IN 1")
Get(T)
Disp T
End
```

Exercice 1

Déterminer ce qu'il se passe lorsque le contact sur le microrupteur est réalisé.

go.eyrolles.com/ti-ch4ex1

Exercice 2

Connecter un second microrupteur et écrire le programme qui permettra de visualiser en permanence l'état des deux microrupteurs.

go.eyrolles.com/ti-ch4ex2

ÉTAPE 2 Connexion du relais et du moteur

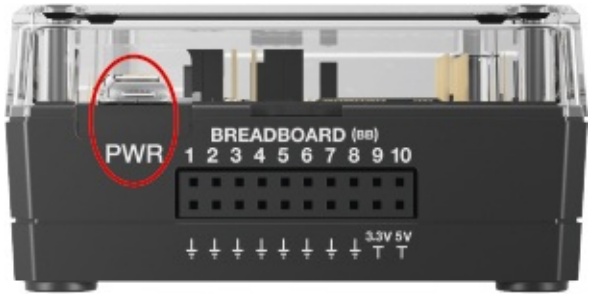
Nous voici donc en mesure de détecter des événements comme la fermeture d'un interrupteur. Nous allons maintenant découvrir comment utiliser un relais pour piloter un petit moteur à courant continu. Ce dernier sera alimenté par une batterie externe, comme déjà signalé.

Outre pour des raisons de sécurité, cet ajout d'alimentation secondaire est utile lorsqu'on souhaite par exemple alimenter une électrovanne en 12 V ou tout autre dispositif nécessitant une tension différente de celle disponible sur le Hub.

Un relais est un interrupteur mécanique dont l'ouverture ou la fermeture est pilotée électroniquement. Dans ce projet, nous allons utiliser un relais de la gamme Grove. Celui-ci possède deux contacts qui sont reliés ensemble lorsqu'il est activé, ainsi qu'une diode rouge servant de témoin lumineux.

Repartons du montage de l'étape 1, enrichi donc d'un second microrupteur, et branchons le relais sur le port OUT 1 du TI-Innovator™ Hub. Il suffit alors d'expliquer au Hub qu'un dispositif de type relais est branché sur ce port. Pour activer le relais, il faudra envoyer la commande **`Send(«SET RELAY 1 TO 1»)`**, et pour le désactiver la commande **`Send(«SET RELAY 1 TO 0»)`**, la diode s'allumant et s'éteignant conjointement.

Il est important de signaler que pour utiliser le relais, nous aurons besoin d'une autre alimentation complémentaire de 5 V. On la branchera directement sur le Hub dans le connecteur PWR prévu à cet effet.



Complétons alors le programme de l'exercice 2 pour que le relais reste activé tant que le microrupteur 1 (celui connecté sur le port BB1) n'est pas enclenché. Si celui-ci est enclenché, alors on désactive le relais.

Saisissez ainsi le programme suivant sur votre calculatrice et testez-le :

```
Send("CONNECT RELAY 1 TO OUT1")
Send("CONNECT DIGITAL.IN 1 TO BB1")
Send("CONNECT DIGITAL.IN 2 TO BB2")
Send("SET RELAY 1 TO 1")
0→T
While T≠ 1
```

```
Send(“READ DIGITAL.IN 1”)
Get(T)
Disp T
End
Send(«SET RELAY 1 TO 0»)
```

Pour l’instant, le second microrupteur n’a pas d’utilité.

Exercice 3

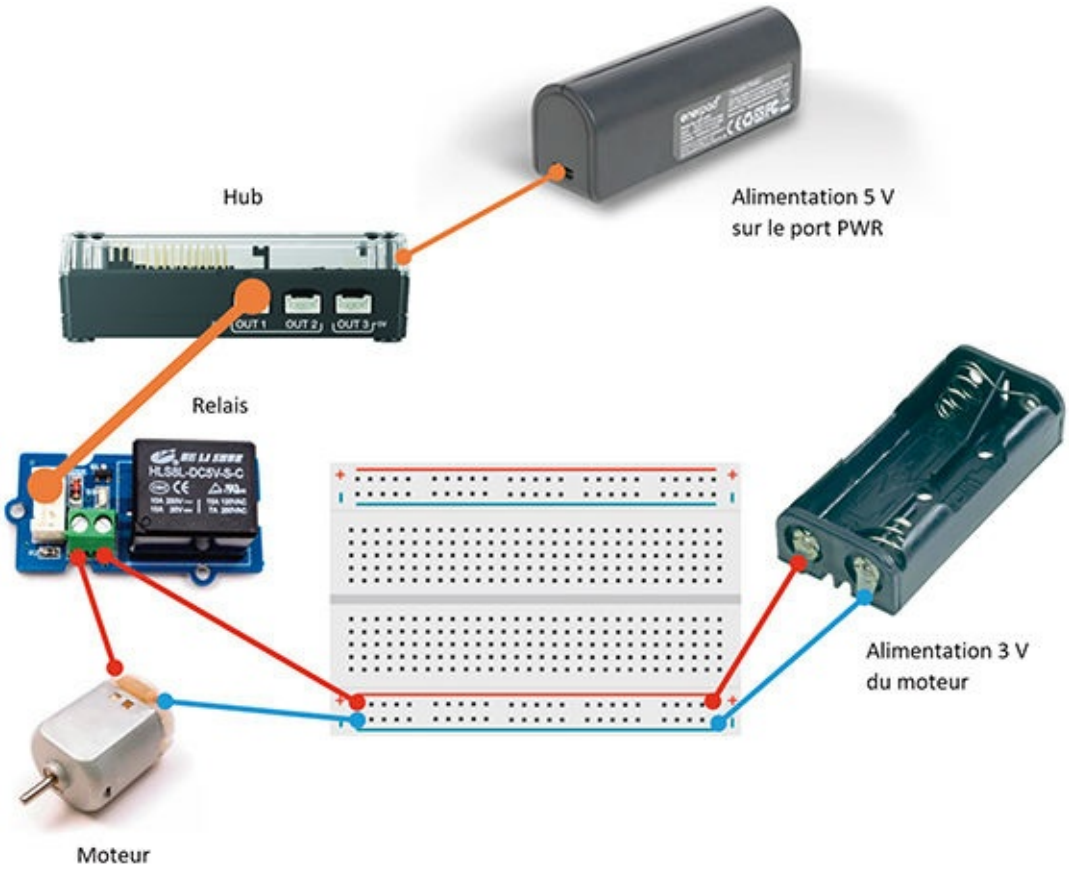
Modifier le programme précédent pour que l’enclenchement du microrupteur 1 active le relais et que l’enclenchement du microrupteur 2 le désactive.

go.eyrolles.com/ti-ch4ex3

Maintenant que nous savons piloter le relais, branchons le moteur à courant continu et son alimentation externe. Dans notre exemple, celui-ci fonctionne à l’aide d’une tension de 3 V, soit deux piles d’1,5 V.

Attention ! Ne pas confondre la batterie externe du moteur avec l’alimentation complémentaire branchée sur le port PWR du Hub, qui elle sert à faire fonctionner le relais.

Sur le schéma ci-après, nous n’avons pas replacé les deux microrupteurs par souci de simplification.



Exercice 4

Inverser les broches de l'alimentation externe du moteur en branchant la broche + sur la ligne bleue de la platine et la broche - sur la ligne rouge. Que se passe-t-il lorsque le programme est exécuté ? Dans quel sens tourne désormais le moteur ?

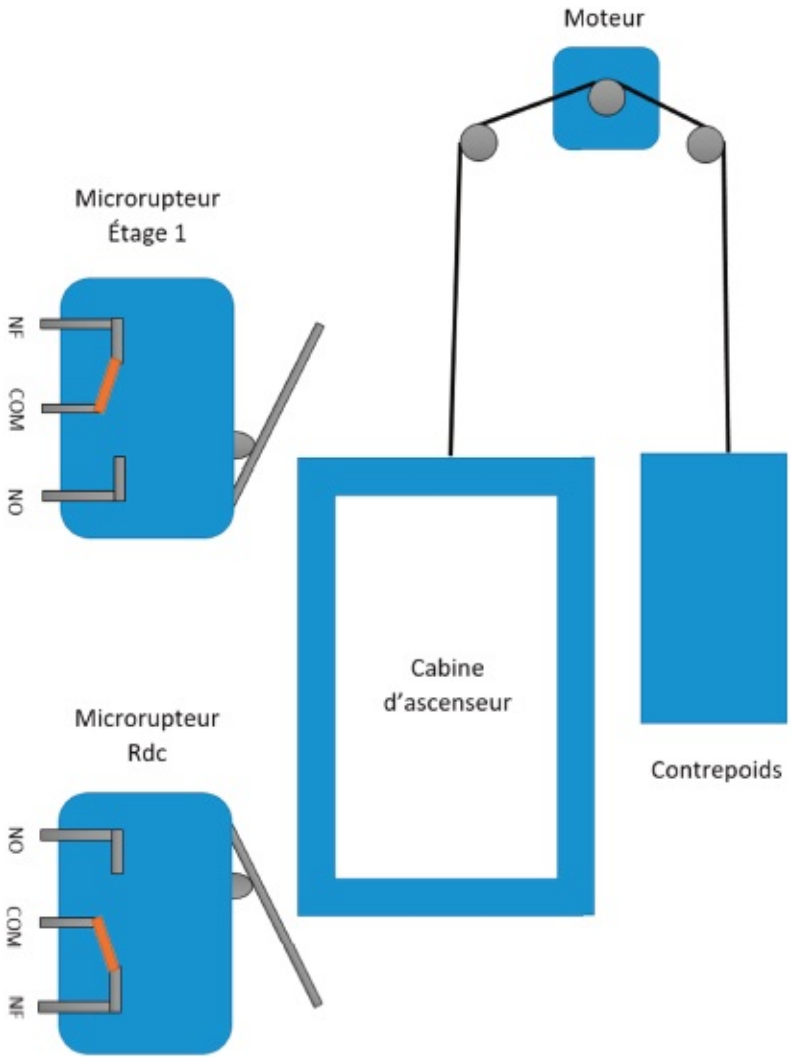
go.eyrolles.com/ti-ch4ex4

ÉTAPE 3 Construction de la cage d'ascenseur

Nous allons à présent fabriquer la cage de notre ascenseur. Pour cela, vous pouvez utiliser une maquette pédagogique en kit comme celle commercialisée par A4 Technologie, sur laquelle vous fixerez vos propres capteurs et actionneurs. Autre solution, vous pouvez choisir de créer vous-même votre modèle, en balsa, en carton, ou à l'aide de jeux de construction en plastique ou en métal.

Sur le principe, votre ascenseur devra utiliser un moteur autour duquel s'enroule un fil. Une extrémité du fil sera reliée à un contrepoids, tandis que l'autre sera raccordée à la cabine de l'ascenseur. La masse du contrepoids devra être sensiblement égale à celle de la cabine.

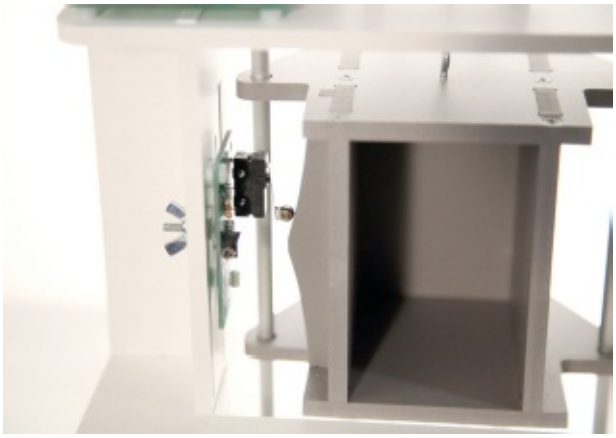
Nous utiliserons deux microrupteurs pour détecter le début et la fin de course de la cabine, qui desservira deux étages. Pour vous aider, voici ci-après un schéma de la cage d'ascenseur.



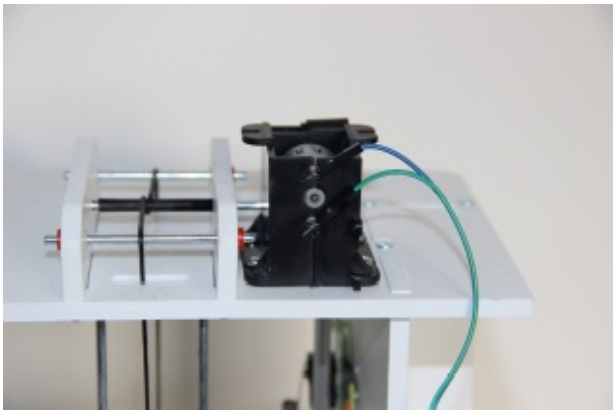
Ici, nous avons opté pour la maquette pédagogique d'A4 Technologie, qui a été légèrement adaptée aux besoins du projet. Sur cette image, on peut voir l'ensemble des éléments qui composent le projet : cabine, contrepoids en arrière-plan, les deux microrupteurs et le moteur.

Cette vue de dos de la maquette permet de mieux comprendre le système de guidage de la cabine et du contrepoids :

On notera la forme astucieuse de l'ergot positionné sur la cabine d'ascenseur. Il permet d'enclencher en douceur et efficacement les microrupteurs de début (Rdc) et de fin (Étage 1) de course.



On voit ici comment nous avons soudé directement les fils sur le moteur en vue de le relier au relais :



Exercice 5

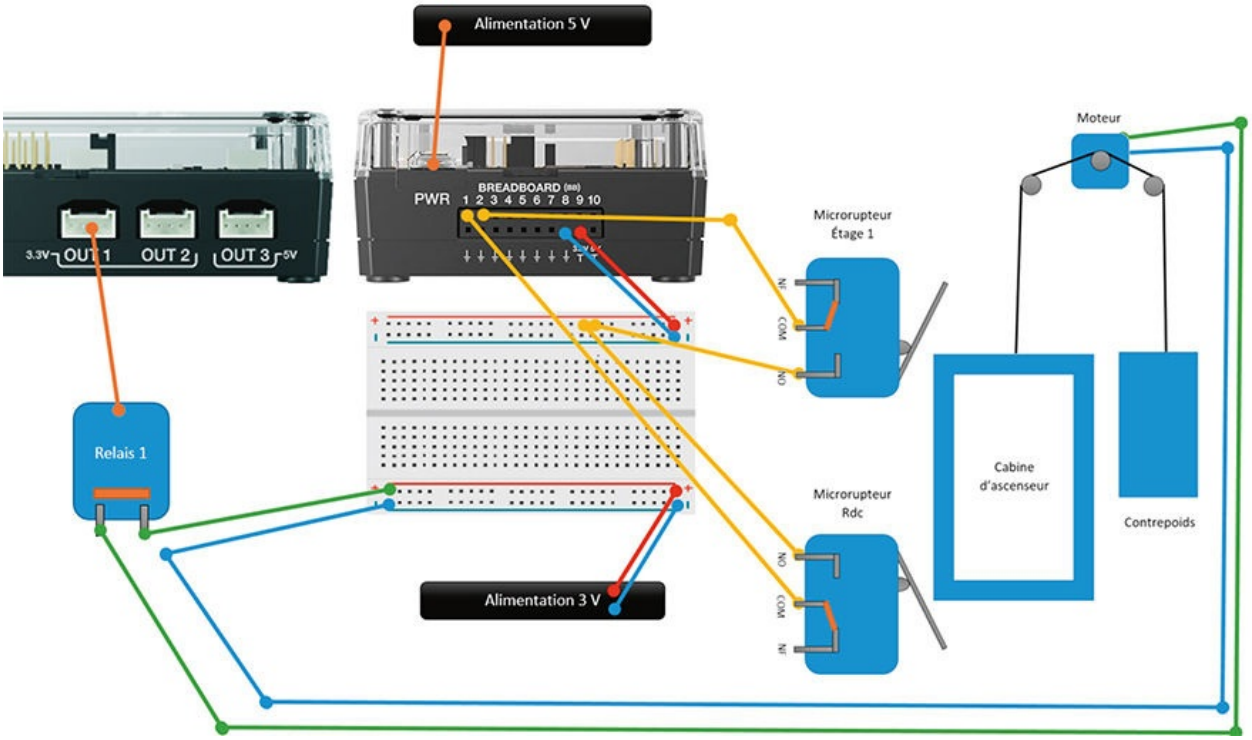
Fabriquer la maquette de la cage d’ascenseur.

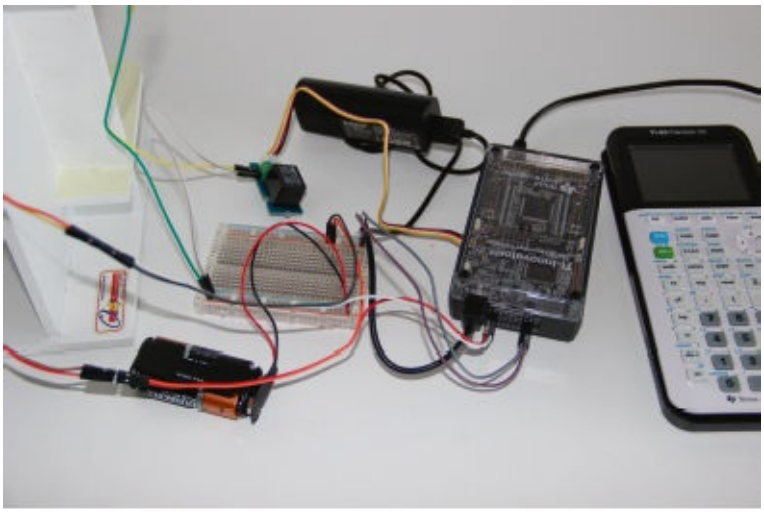
ÉTAPE 4 Branchement du circuit et écriture du code

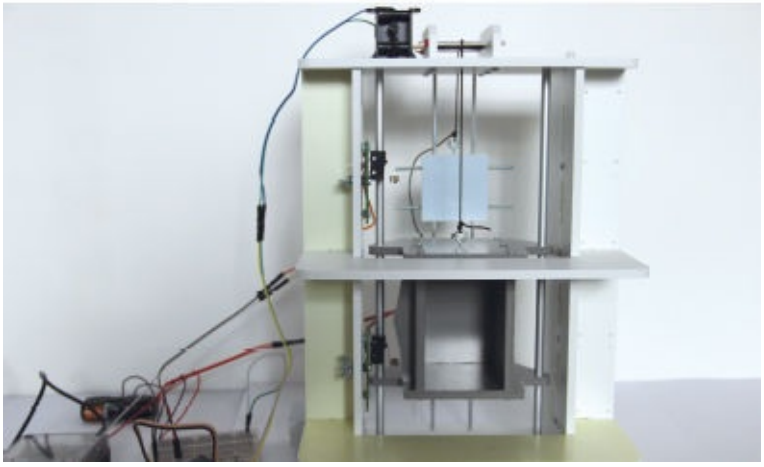
Il reste maintenant à connecter les différents éléments du projet et à relier l’ensemble à la calculatrice.

Comme décrit à l’étape 1, il faut brancher le microrupteur 1 sur le port BB1 du

TI-Innovator™ Hub, et le microrupteur 2 sur le port BB2. Le relais sera relié au port OUT 1 du Hub, tandis que le moteur de l'ascenseur devra être connecté selon le schéma de branchement de l'étape 2. On placera la cabine en position basse. Bien sûr, on n'oubliera pas de brancher le Hub à la TI-83 CE Premium et l'alimentation externe sur le port PWR.







Nous allons maintenant créer un programme qui activera le relais une fois que l'utilisateur aura appuyé sur la touche 1 (code 92) de la calculatrice. La cabine d'ascenseur entamera lors sa montée et s'arrêtera automatiquement lorsque le microrupteur 2 aura été enclenché par le contact avec la cabine. Une pression de la touche entrer (code 105) arrêtera le programme.

En prévision de l'étape 5, nous vous proposons de créer en réalité deux programmes : un programme principal nommé **PRINCIP** gérant les pressions sur les touches de la calculatrice et un sous-programme **MONTEE** gérant la montée de l'ascenseur.

Exercice 6

Compléter les pointillés du programme `PRINCIP` pour qu'il réalise les actions décrites précédemment

```
Send("CONNECT …… …… TO OUT1")
Send("CONNECT …… …… TO BB1")
Send("CONNECT …… …… TO BB2")
0→K
Disp «APPUYER SUR 1»
Disp «ENTRER POUR SORTIR»
While K≠ ……
getKey→K
If K=……
……
End
```

go.eyrolles.com/ti-ch4ex6

Exercice 7

Compléter les pointillés du programme `MONTEE` pour qu'il réalise les actions décrites précédemment.

```
0→T
Send("READ DIGITAL.IN 2")
Get(T)
If T=0
Then
Send("SET RELAY 1 TO ……")
Disp "MONTEE"
While T≠1
Send("READ DIGITAL.IN 2")
Get(T)
End
Send("SET RELAY 1 TO ……")
Disp "ETAGE 1"
End
```

Il n'est pas nécessaire de faire appel à la fonction `CONNECT` puisqu'elle a déjà été appelée par le programme `PRINCIP`.

go.eyrolles.com/ti-ch4ex7

Exercice 8

Expliquer pourquoi le programme `MONTEE` commence par tester l'état du microrupteur 2.

go.eyrolles.com/ti-ch4ex8

ÉTAPE 5 Gestion de la descente

À ce stade du projet, vous disposez donc d'une cage d'ascenseur capable de partir du rez-de-chaussée pour arriver au premier étage. Nous vous proposons à présent de compléter et modifier judicieusement les programmes précédents pour réaliser un ascenseur gérant la montée et la descente de la cabine.

Comme vous vous doutez, vous devrez créer un deuxième sous-programme nommé **`DESCENTE`**. Il faudra également inclure l'appel de ce sous-programme dans le programme principal.

DÉFI 1

Le défi se situe dans la gestion de l'alimentation du moteur. Plus précisément, il vous faudra trouver une méthode pour inverser électroniquement le sens de rotation du moteur. De nombreuses solutions existent. Nous en avons retenu une. Saurez-vous mettre la vôtre en œuvre ?

Vous voici arrivé à la fin de ce livre. Grâce aux techniques qui vous ont été

présentées, vous êtes maintenant en mesure de programmer vos propres jeux et construire vos propres robots. Laissez votre créativité s’exprimer !

Liens utiles

- **https://education.ti.com/fr/france/home**

Vous trouverez sur ce site toutes les informations pour mettre à jour votre TI-83 Premium CE, télécharger le logiciel TI Connect™ CE et accéder à des tutoriels sur la programmation de votre calculatrice.

- **https://education.ti.com/fr/purchase/purchase_dealers**

Si vous souhaitez vous procurer une calculatrice TI-83 Premium CE ou un TI-Innovator™ Hub, cette page vous indiquera la liste des principaux fournisseurs.

- **http://www.lestutosmaths.fr/home**

Rendez-vous sur ce site pour maîtriser encore davantage votre calculatrice TI-83 Premium CE. Il contient beaucoup de vidéos et de tutoriels.

- **https://education.ti.com/fr/customer-support**

En cas de problème avec votre TI-83 Premium CE ou votre TI-Innovator™ Hub, consultez ce site.

Pour suivre toutes les nouveautés numériques du Groupe Eyrolles, retrouvez-nous sur Twitter et Facebook

@ebookEyrolles

EbooksEyrolles

Et retrouvez toutes les nouveautés papier sur

@Eyrolles

Eyrolles